COMPUTER SECURITY
CS419

GROUP PROJECT DETAILS

GROUP PROJECT

TEAM PROJECT

- Team leaders have been announced on Piazza
- Team leaders please add your members to Google Doc
- Team members will confirm you are added
- If you are still looking for a team, you can use Piazza or the Google Doc to signup
- If you do not participate team project, you fail in this class.
- You can start now

GRADING

- Basic: 10 points
  - A functional product implementing the basic idea
  - Everyone gets the same for this part
- Bonus: up to 20 points in total for a group
  - Divided among members based on the final report
  - Notice in your final report, you should list contributions of individuals and the percentage of shared bonus

GENERAL INFORMATION

- Programming Languages
  - You can free to use your favorite programming language.
- Platform
  - For System security, Linux.
  - For others, you can use Windows/Mac/Linux.
- GUI
  - You are strongly ***recommended*** to have an easy to use GUI for Crypto and ML

TOPICS

- Crypto
- Software Security
- System Security
- Machine Learning

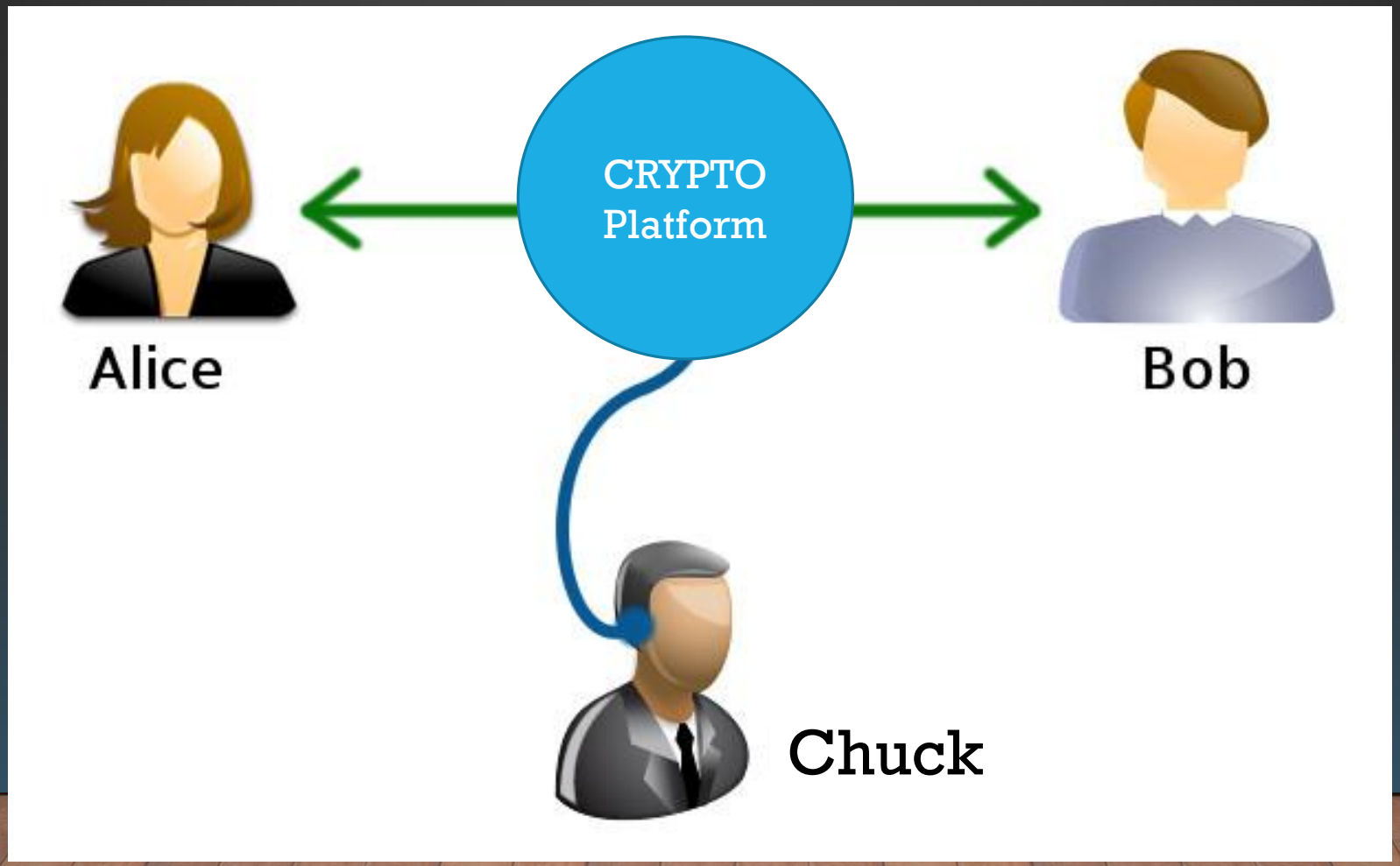
CRYPTO GAME

- A crypto platform that allow users/attackers to encrypt and decrypt
- Two users: Alice and Bob
  - Two modes: shred key and PKE
- One attack: Chuck
  - Four modes: ciphertext-only, know-plaintexts, chosen-plaintext, chosen-ciphertext
- Can reuse some existing libraries
- Must implement at least 3 ciphers by yourself
  - Can NOT include shift cipher

MINIMAL

- A platform with Alice and Bob as users, and Chuck as the attack
- Support 4 modes
- Support 3 self-implemented algorithms with at least one public/private key system
- Support brute force attack

CRYPTO



ALICE AND BOB

- Can send/receive information
- Can generate its own key using the provided system
  - Alice and Bob can get keys from the system and then start to communicate
- The system can provide the same key to Alice and Bob (only!)
- The system can provide public and private keys to the owner
- … but does not leak private key to the other party

CHUCK

- Can query the crypto system in for modes:
  - ciphertext-only
  - know-plaintexts
  - chosen-plaintext
  - chosen-ciphertext
- Tries to break the system by guessing what Alice and Bob said
  - Basic toolbox: brute force attack, frequency analysis etc.
- Keys are always hidden to Chuck

CRYPTO SYSTEM

- Supports what Alice/Bob and Chuck can do
- Provide basic algorithms for Alice and Bob to use, 3 of them have to be implemented by you from scratch
- Can re-use existing libraries to implement more algorithms
  - E.g., RC4, DES, AES
- Needs to support all types of cipher types (at least one algorithm in each category) in class

WHAT IS MORE…

- Supporting strangers to join the communication
  - When Emma joins the system, she does not know anyone and wants to talk to Alice.
  - Emma needs to establish trust with Alice.
  - Alice and Emma trusts the system.
  - Alice and Emma cannot trust each other.
  - Need a key exchange and agreement protocol
- Anything you can think of
  - E.g., customizing your own algorithm
  - E.g., multi party communication

QUESTIONS?

SOFTWARE SECURITY

- Fuzzing with AFL (American Fuzzy Loop)
  - http://lcamtuf.coredump.cx/afl/
- Improve AFL by any means
  - Seed selection, using metrics other than coverage etc.
- Test on LAVA-M and Google test suites
  - http://panda.moyix.net/~moyix/lava_corpus.tar.xz
  - https://github.com/google/fuzzer-test-suite
- Compare AFL with your improved version

MINIMAL

- Implementing at least one improvement
- Tests on LAVA-M and Google Test Suite
  - AFL and improved version

FUZZING

- 1988, Barton Miller, the father of fuzz testing
  - Proposed the concept of fuzzing: use random data to test programs till it crashes
- 2001, PROTOS, applied fuzzing on network protocols
  - *https://www.ee.oulu.fi/research/ouspg/Protos*
  - Later it became Codenomicon (the discovery of Heartbleed bug)
- 2002, BlackHat USA, Immunity published SPIKE
  - Test border conditions
  - Customized definition for network protocol fuzzing

FUZZING

- Peach Fuzzer
  - https://www.peach.tech/
  - File based fuzzing
  - Fuzz file formats, network protocols, ActiveX
  - Still being used: https://github.com/aflsmart/aflsmart

FUZZING BROWSERS

- 2008, Mozilla funfuzz
  - Jsfunfuzz
  - DOMfuzz
  - https://github.com/MozillaSecurity/funfuzz
  - Based on PL grammars to generate inputs
- Many more on browsers
  - Dharma, domato, grinder, nduja, crossfuzz
  - SQL is included! Because browsers use databases like SQLite
- PDF fuzzers use similar techniques

COVERAGE BASED FUZZING

- 2013, AFL uses instrumentation + QEMU to implement coverage guided fuzzing, fuzzing enters a new decade
- AFL starts to be popular in 2014, 2015
- Why? A lot of CVEs
- AFL based fuzzing
  - Winafl, libfuzzer, AFLFast, Vuzzer, syzkaller,
  - Fuzzing kernels, Go, Pythong, JS, Ruby, Windows, IoT, Android … …

THE VERY HIGH-LEVEL IDEA OF AFL

- Start with one seed input
- Test the program
  - Gather coverage information
- Randomly generate new inputs
- Select new inputs to test the program
- Just repeat

WHERE CAN WE IMPROVE?

- How to generate new inputs?
  - Purely random? Which bit to change?
- How to get high coverage?
  - Some conditions are just harder to satisfy
- How to select the next seed input which may lead to high coverage?
  - Power spent on mutating one seed vs. multiple
- Why does it have to be coverage? Does it reflect everything?
- How can we speed it up?

SOME EXISTING WORK YOU MAY WANT TO CHECK OUT

- Driller
  - AFL + symbolic execution
- AFLFast
  - AFL + Markov chain
- AFFSMART
  - AFL + Peach

REQUIREMENTS

- You can re-use tools like KLEE (for symbolic execution) etc.
- You cannot re-use existing fuzzers like Driller
- How to evaluate fuzz testing?
  - https://www.cs.umd.edu/~mwh/papers/fuzzeval.pdf

QUESTIONS?

SYSTEM SECURITY

- A protected file system
- For a given folder and files inside, the system only allows the account Alice to use certain programs to create/read/edit/delete it
- You need to assign correct permissions
- Other accounts are not able to read the content
- Purely user level file system is fine

MINIMAL

- Supporting file creation/read/write/deletion for Alice only in a specific folder
- Root user cannot get the file content

LINUX PERMISSIONS

- We will cover this topic right after crypto related topics
- Linux has permissions to isolate accesses
  - Alice can be (almost) the only one to read/write/delete/create files
- What is the challenge?
  - ROOT user
  - Root accounts can revert all protections you may add to the system

HOW DO WE SOLVE IT?

- There are two ways of protection
  - Hide the existence → Access control
  - Hide the content → Crypto
- Crypto introduces new challenges: performance
  - If edit is supported, how do we encrypt?
  - Where do we store the key?
- If cannot fully prevent it, audit the modification
  - Linux audit

REQUIREMENT

- As long as your implemented system prevents the other users as well as root from reading the real content
- There are many possible decision choices
- TEXT files only
  - There is no need to implement a text editor (although, you can enhance simple ones by adding more functionalities)
  - You can use a wrapper, which calls other editors

WHAT DO YOU EXPECT

- Please explain how you encrypt and decrypt files
- Please explain how do you deal with random access/edit
- Please explain the trade-off in the design
- Please explain how do you store keys
- Please explain how do you authenticate Alice
- Please explain the provided security features

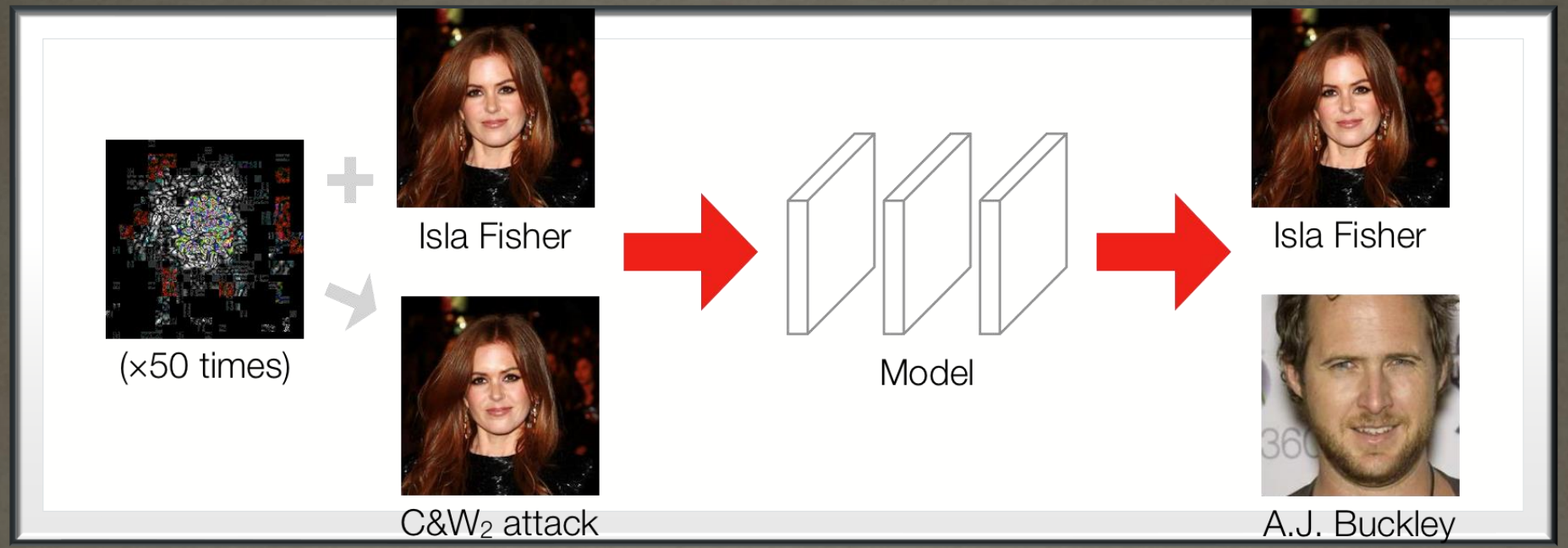
QUESTIONS?

ML SECURITY

- A platform for adversarial attack and defenses
- Administrator can publish datasets to users to train models
  - MNIST, CIFAR-10
- Users train robust models
- Users submit adversarial examples to attack all others' models
- A leaderboard GUI is required to show the accuracy of each model and attack success rate
- You can use existing implementations of many attacks/defenses, but one attack and one defense have to be your own implementation

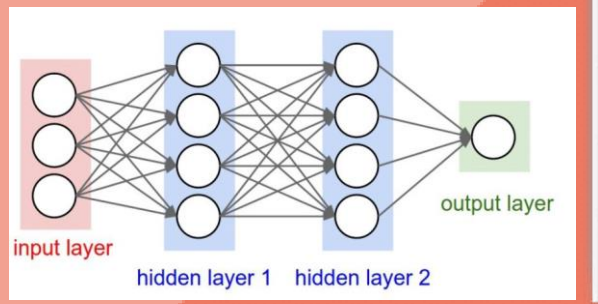
MINIMAL

- Built-in datasets, MNIST and CIFAR
- Built-in models (at least one for each dataset)
- Up-to-date leaderboard
- At least one attack algorithm and one defense



ADVERSARIAL EXAMPLE

36



Rank	Member Name	Reputation Points	Badge	Member Since
1	Melissa	189		4 months ago 4/21/2016
2	Seth Waterfall	115		4 months ago 4/21/2016
3	Alexandra Landeta	77		4 months ago 4/26/2016
4	Ed Viesturs	61		4 months ago 4/21/2016
5	Lenard Johnson	37		4 months ago 4/26/2016
6	Pengcheng	20		4 months ago 4/26/2016
7	Barbara	18		4 months ago 4/26/2016
8	Zoe	11		1 month ago 8/3/2016

- A dataset users can download
- A leader board

ATTACK MODES

- White-box
  - Administrator publishes the model, users can also publish models
  - Users attack it with full control
- Black-box
  - Administrator publishes the dataset only
  - Users attack each other's model (Adversarial examples transfer!)
- Gray-box
  - Administrator publishes the model with constraints
    - # query or frequency, output label or result vector
- Administrators configure these modes and also choose these options

USER

- Users can perform attacks
  - Submit attack images
  - Submit attack algorithm
- Users can train robust models
  - Submit the trained models
- The system can
  - Automatically attack published/submitted models using submitted attack images/algorithms
  - Update the leaderboard

QUESTIONS?

WHAT IS THE PRODUCT?

- Artifacts
  - Code
  - Documentation including dependencies, compilation instructions and parameters, inputs to program etc.
  - A report including your detailed design, evaluation
- Presentation in the last week!
  - Live demo is required.

QUESTIONS?