Operating Systems!

Security:
Introduction to (OS) Security
(part 2)

Prof. Travis Peters

Montana State University

CS 460 - Operating Systems

Fall 2020

https://www.cs.montana.edu/cs460

Some diagrams and notes used in this slide deck have been adapted from Sean Smith's OS courses @ Dartmouth. Thanks, Sean!

Today

- Announcements
  - Exam 2 details coming soon (FYI similar format as Exam 1)

- Upcoming Deadlines
  - **Project Submission (HARD DEADLINE)**
    Sunday [11/15/2020] @ 11:59 PM (MST)
  - **Project Evaluations (HARD DEADLINE)**
    Wednesday [11/18/2020] @ 11:59 PM (MST)
  - **Course Evaluations!**
    Wednesday [11/18/2020] @ 11:59 PM (MST) -> ~17%
  - **Exam #2**
    -> Moved to Friday (11/20)

Today

- Agenda
  - ~~The Basics~~
  - Buffer Overflows (an illustrative example of [OS] vulns and countermeasures)
  - Access Control
  - (Authentication)
  - (The Future of Authentication?)
  - (SGX & Trusted I/O?)

-> Take CSCI 476/594 next semester! ☺
(Tu/Th @ 3:05-4:20pm) -> I already see almost 60 people signed up!

Buffer Overflows

Can you give me an idea of how attackers exploit systems?
How could an attacker, for example, run arbitrary code on a machine?

The Buffer Overflow: Basic Ideas

An example of a common exploit technique!

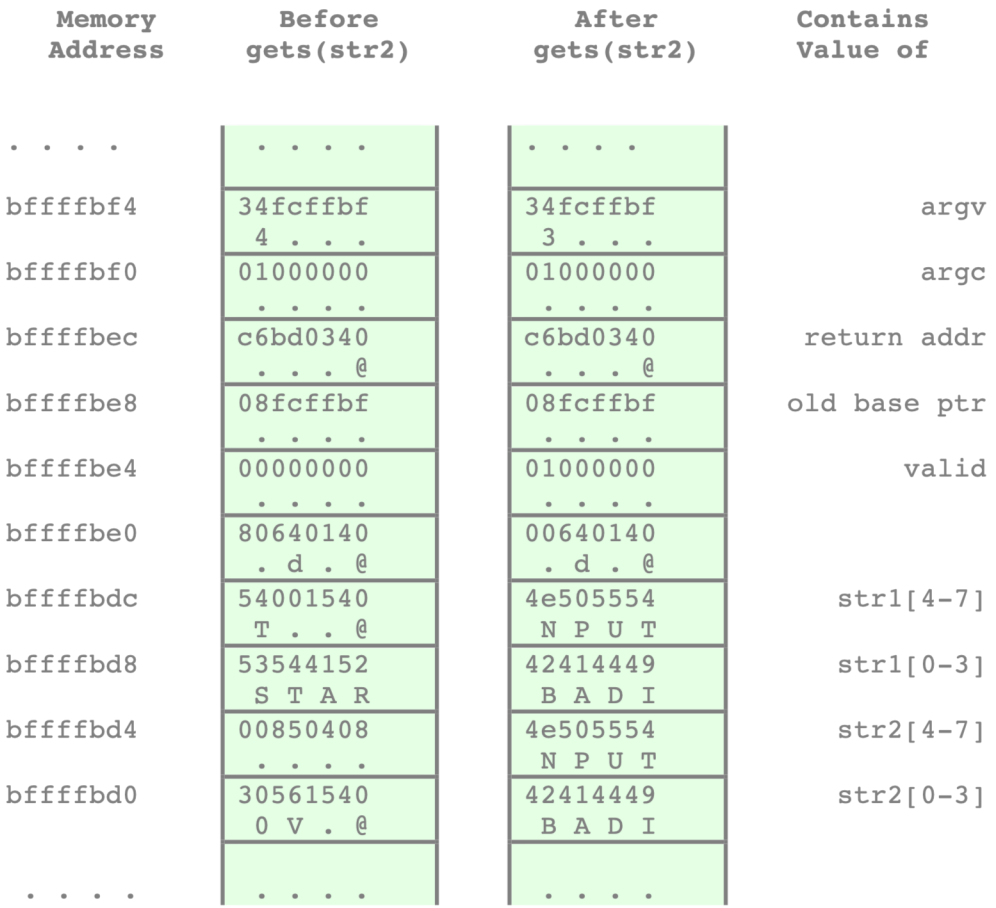
```
int main(int argc, char *argv[]) {
    int valid = FALSE;
    char str1[8];
    char str2[8];

    next_tag(str1);
    gets(str2);
    if (strncmp(str1, str2, 8) == 0)
        valid = TRUE;
    printf("buffer1: str1(%s), str2(%s), valid(%d)\n", str1, str2, valid);
}
```

(a) Basic buffer overflow C code

```
$ cc -g -o buffer1 buffer1.c
$ ./buffer1
START
buffer1: str1(START), str2(START), valid(1)
$ ./buffer1
EVILINPUTVALUE
buffer1: str1(TVALUE), str2(EVILINPUTVALUE), valid(0)
$ ./buffer1
BADINPUTBADINPUT
buffer1: str1(BADINPUT), str2(BADINPUTBADINPUT), valid(1)
```

(b) Basic buffer overflow example runs



-> walk through Sean's buffer overflow sequence for more details!

The Buffer Overflow: Defenses / Countermeasures

An example of a common exploit technique!

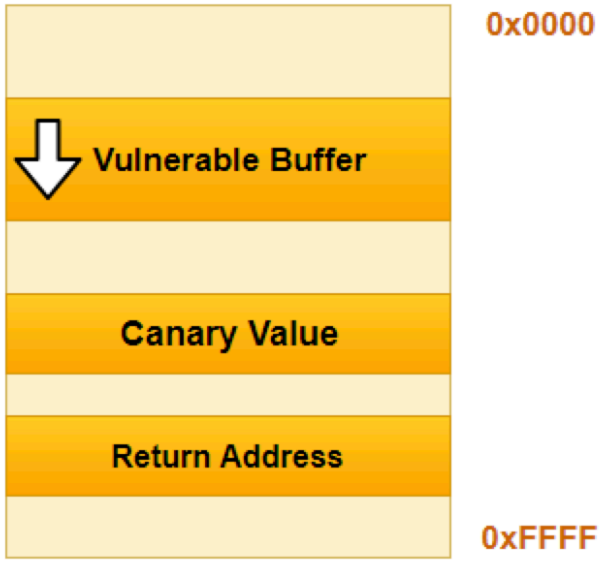
Compile-time defenses

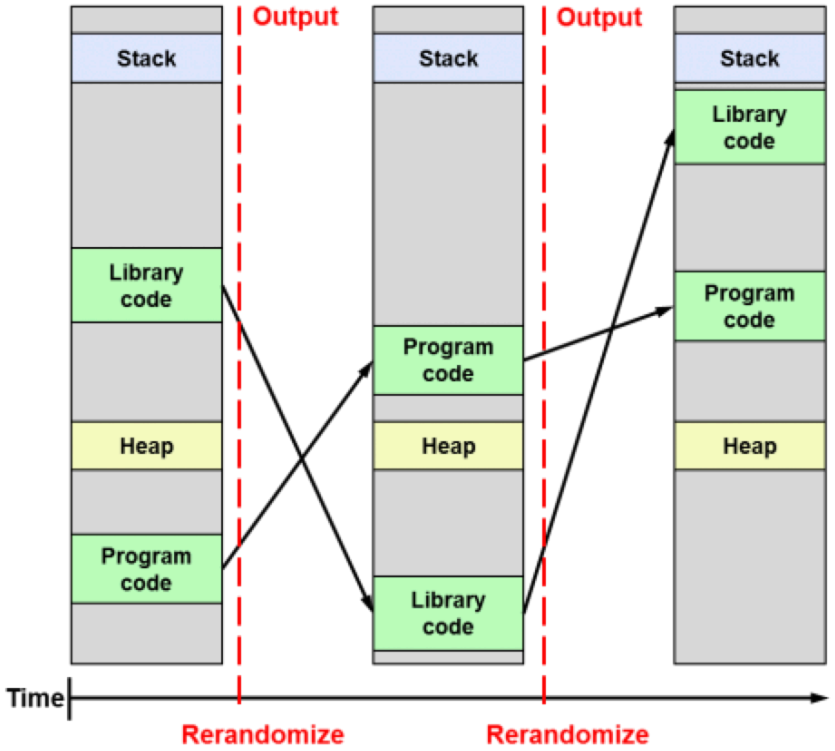
- high-level languages
- language extensions (e.g., libsafe)
- static analysis
- stack guards ("canaries") (shown below)

vs.

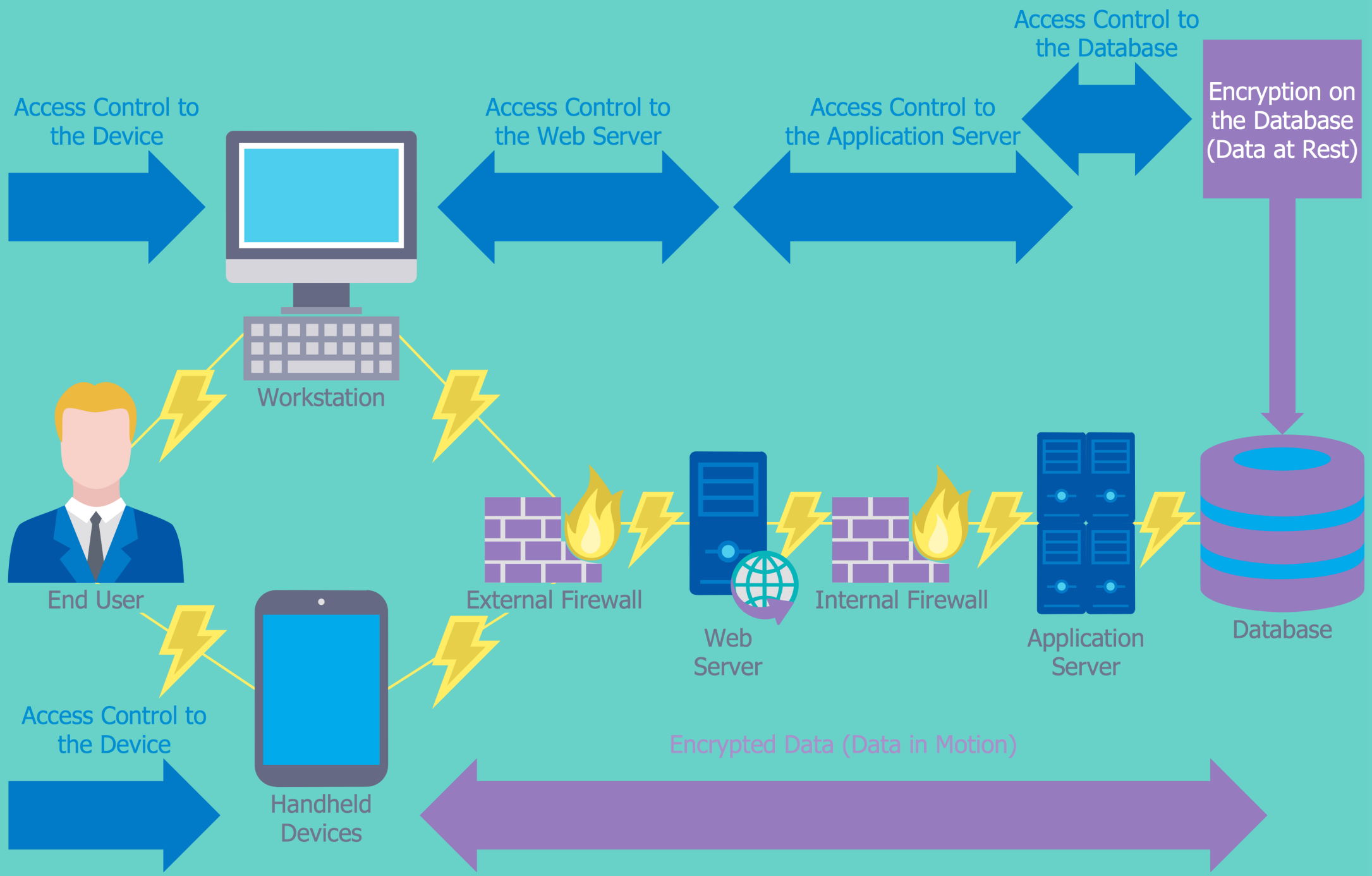
run-time defenses

- non-executable address space protection (nx-bit on x86)
- guard pages
- ASLR (shown below)





Access Control


Access Control to the Database
Access Control to the Device
Access Control to the Web Server
Access Control to the Application Server
Encryption on the Database (Data at Rest)
Workstation
End User
External Firewall
Web Server
Internal Firewall
Application Server
Database
Access Control to the Device
Handheld Devices
Encrypted Data (Data in Motion)

Access Control:

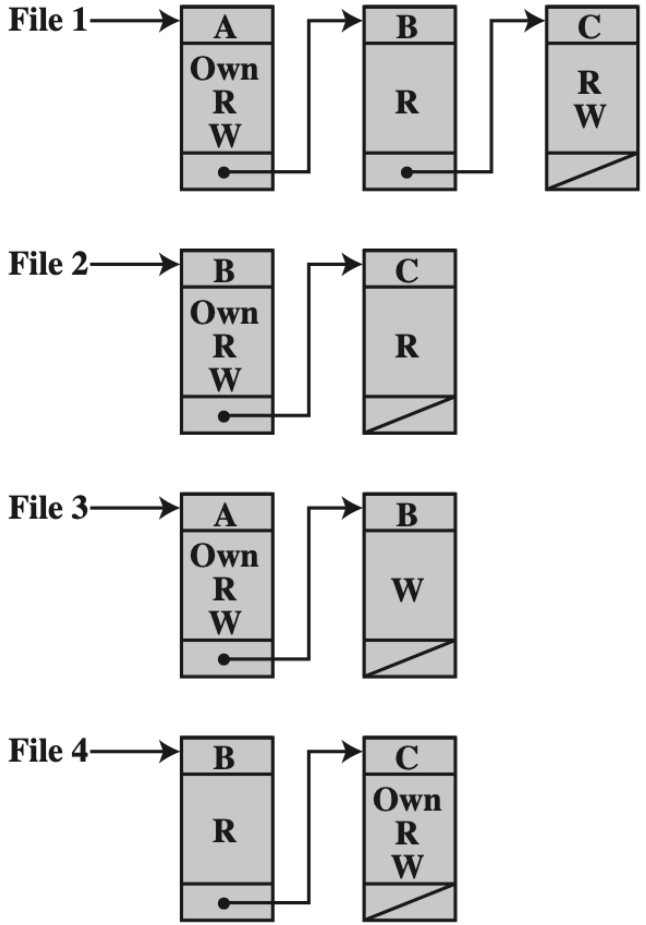
(File System) Access Control Matrix

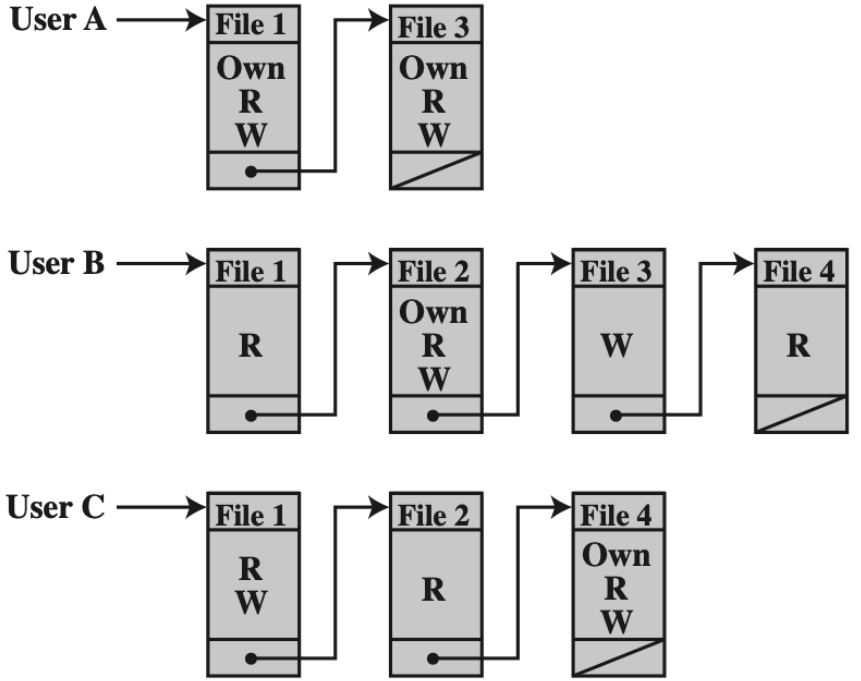
		OBJECTS			
		File 1	File 2	File 3	File 4
SUBJECTS	User A	Own Read Write		Own Read Write	
	User B	Read	Own Read Write	Write	Read
	User C	Read Write	Read		Own Read Write

Access Control:

Access Control List (ACL)

		OBJECTS			
		File 1	File 2	File 3	File 4
SUBJECTS	User A	Own Read Write		Own Read Write	
	User B	Read	Own Read Write	Write	Read
	User C	Read Write	Read		Own Read Write





Access Control:

Unix File Modes & Permissions

- Every Unix file has a set of **permissions** that determine whether someone* can read, write, or run the file.
  - Try: `ls -l ~`
  - Try: `ls -l /dev`

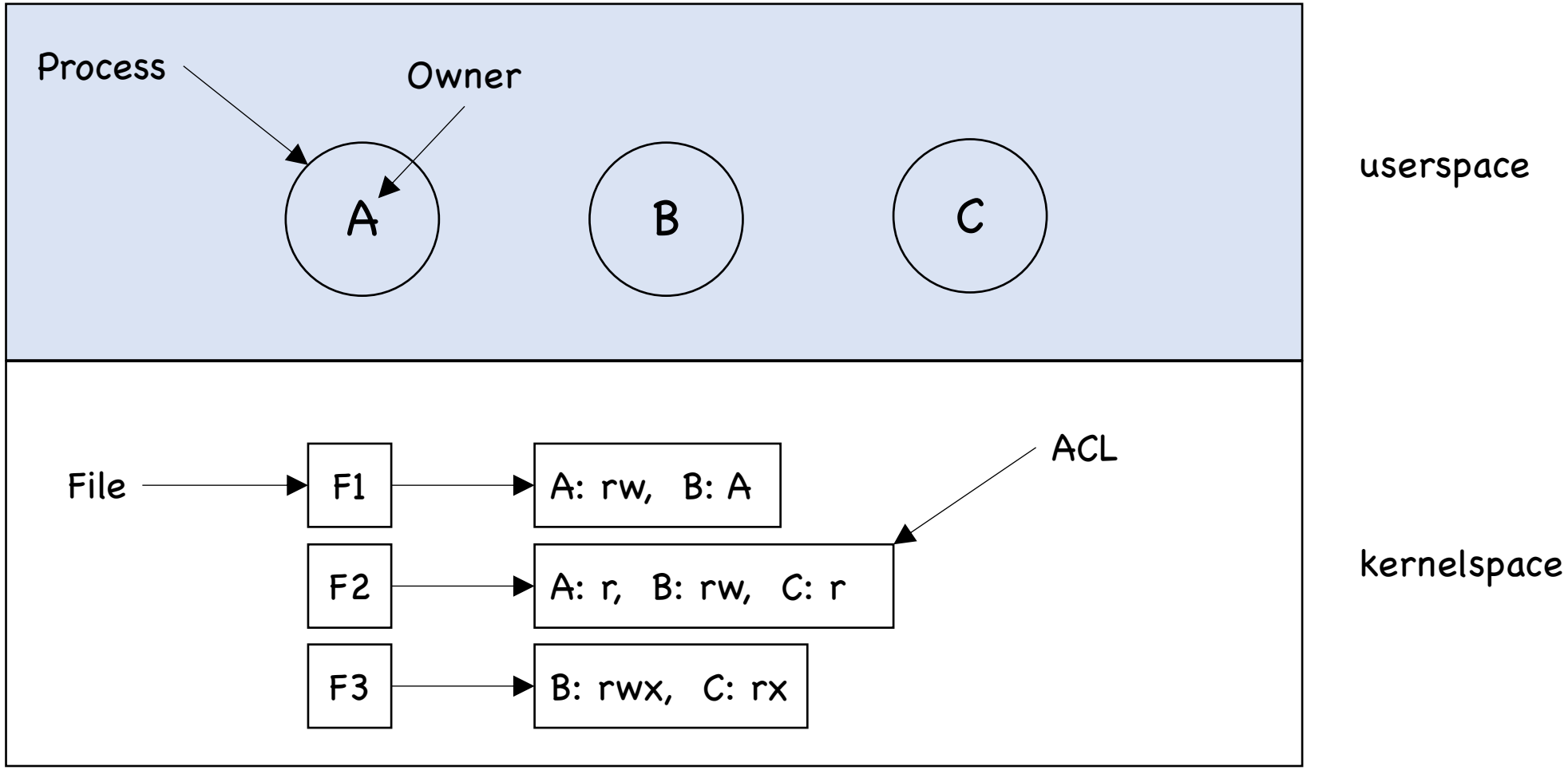
- File mode (4 parts)
  -> [file type][user][group][other]
  - file type
  - ugo
  - rwx

```
$ls -l file
-rw-r--r-- owner group ... file
```

- How do we change these settings?
  - Ex: allow other members of my **group** to **write** "file"?

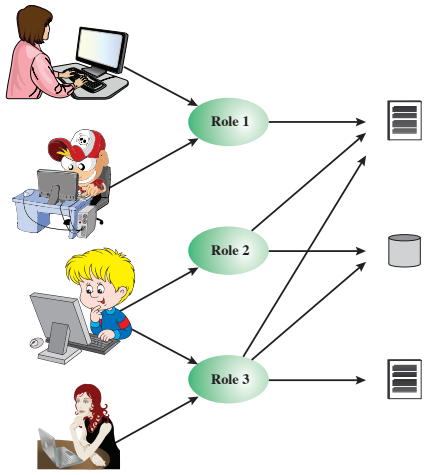
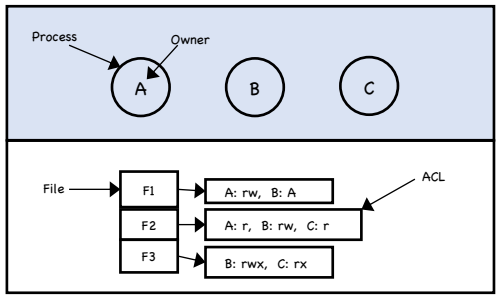
Access Control:

A Unix ACL Example



Access Control:

Variations

DAC	MAC	RBAC	ABAC
Each subject can perform action(s) on objects	Each subject can perform action(s) on objects	Roles have useful meaning and context for access permissions	Policies expressed as a complex Boolean rule set that can evaluate many different attributes
subjects & objects each have associated security attributes	subjects & objects each have associated security attributes	Example: student, lab manager, professor, dean, president	Examples:
Permissions based on user/group	Permissions based on user/group	Each **subject** is mapped to a **role(s);** each role has **permission** to perform **actions** on **objects.**	**Subject Attributes** (age, clearance, dept. role, job title)
Discretionary -> users can grant others access similar to its own permissions	**Mandatory ->** sec. policy is centrally controlled; users do not have the ability to override the policy (e.g., cannot grant others access to files that would otherwise be restricted)		**Action Attributes** (read, write, view, approve)
			**Resource Attributes** (object type, sensitivity, location)
			Contextual Attributes (time, location)
Subject can Action to Object Subject can grant other Subject	Subject can Action to Object Object can be Action by Subject	Subject is a Role which has Permission of Action to Object	Subject who is xxx can Action to Object which is xxx in Environment

Access Control:

Looking Forward…

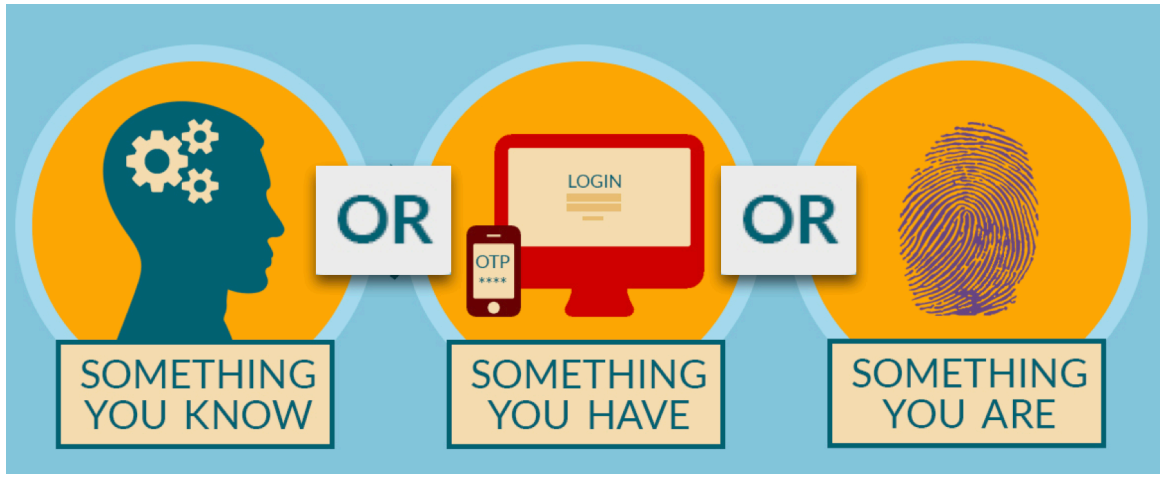
- **Basic Access Control**
  - UNIX, ACL, various capability systems
- **Aggregated Access Matrix**
  - TE, RBAC, groups and attributes, parameterized
- **Plus Domain Transitions**
  - DTE, SELinux, Java
- **Lattice Access Control Models**
  - Bell-LaPadula, Biba, Denning
- **Predicate Models**
  - ASL, OASIS, domain-specific models, many others
- **Safety Models**
  - Take-grant, Schematic Protection Model, Typed Access Matrix
- **Issues**
  - Revocation of access, implementation complexity, usability, …

EXTRAS!

Authentication

What is authentication?

- **Identification**
  Determining who someone is
  -> Who are you?

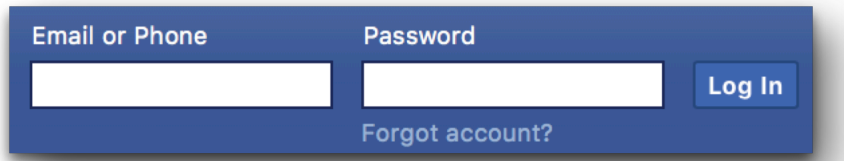


- **Authentication**
  How one proves that they are who they say they are
  -> Who are you? Prove it!

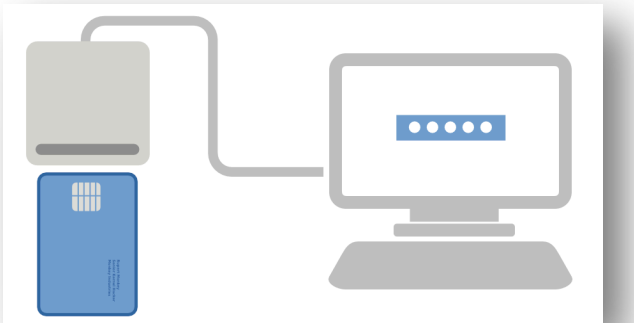
E.g.,
Identification says "I'm Travis!" (name, username, etc.)
Authentication proves it with something (password, birth certificate, etc.).

Forms of authentication

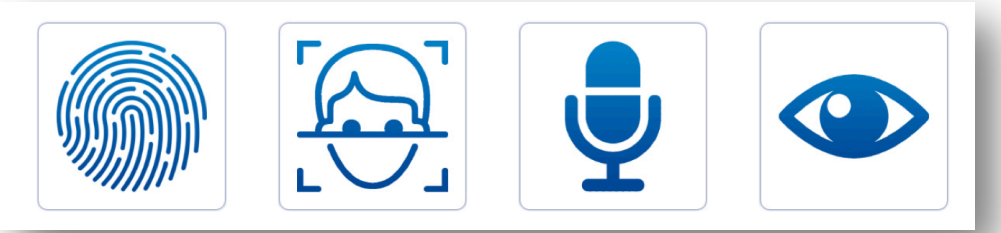
username/password



token



biometrics

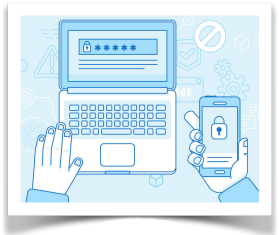


multi-factor authentication

MFA (e.g., 2FA) Is Great… So Problem Solved, Right?

Opinion

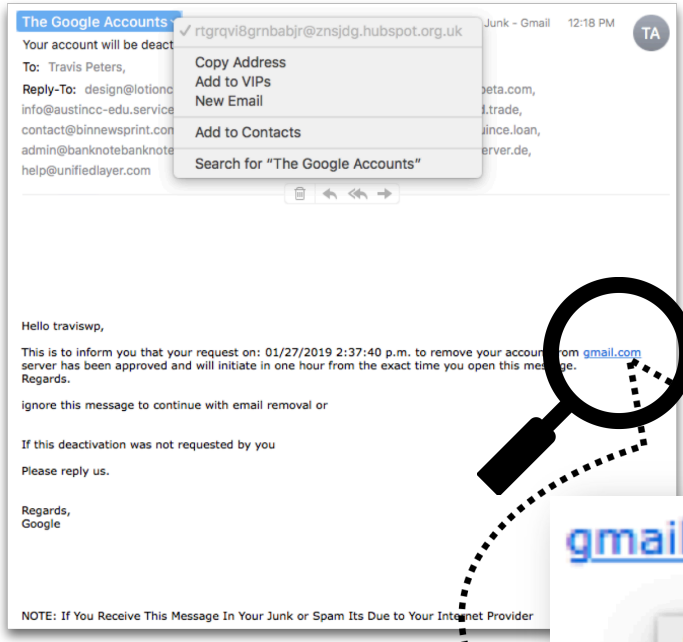
Two-Factor Authentication Might Not Keep You Safe



The online security "best practice" is still vulnerable to phishing attacks.

Setting up phishing websites that look like the login pages for well-known web services is a common way to steal passwords online. Here's the way it works: Someone designs a website that looks almost exactly like Bank of America's website and then emails you a message, purporting to be from Bank of America, warning you that your account is about to expire, or your information needs to be updated, and directing you to a fake site where you believe you're logging into your bank account but instead are just typing your password into a website owned by scammers.

you're the hacker, all it takes is adding a component to your fake Bank of America website so that after you prompt someone for his password, you try to log in to his real Bank of America account using the password he has just provided, triggering a second-factor alert that doesn't alarm him because he thinks he's signing into Bank of America too. Then, on your



What does authentication look like today?

A subset of authentication materials
carried by one participant

S. Mare et al. A Study of Authentication in Daily Life,
USENIX Usable Privacy and Security 2016.

- **Auth Materials**
  Car key, house key, RFID badge, bike key, RSA token, bus pass, credit card, driver's license, …
- **Auth Targets**
  Doors, computers, services, …
- **State-of-the-art**
  Key fobs/tokens,
  Biometrics (Fingerprint, Face ID),
  Password Managers, …

Activity: What Makes a Good Password?

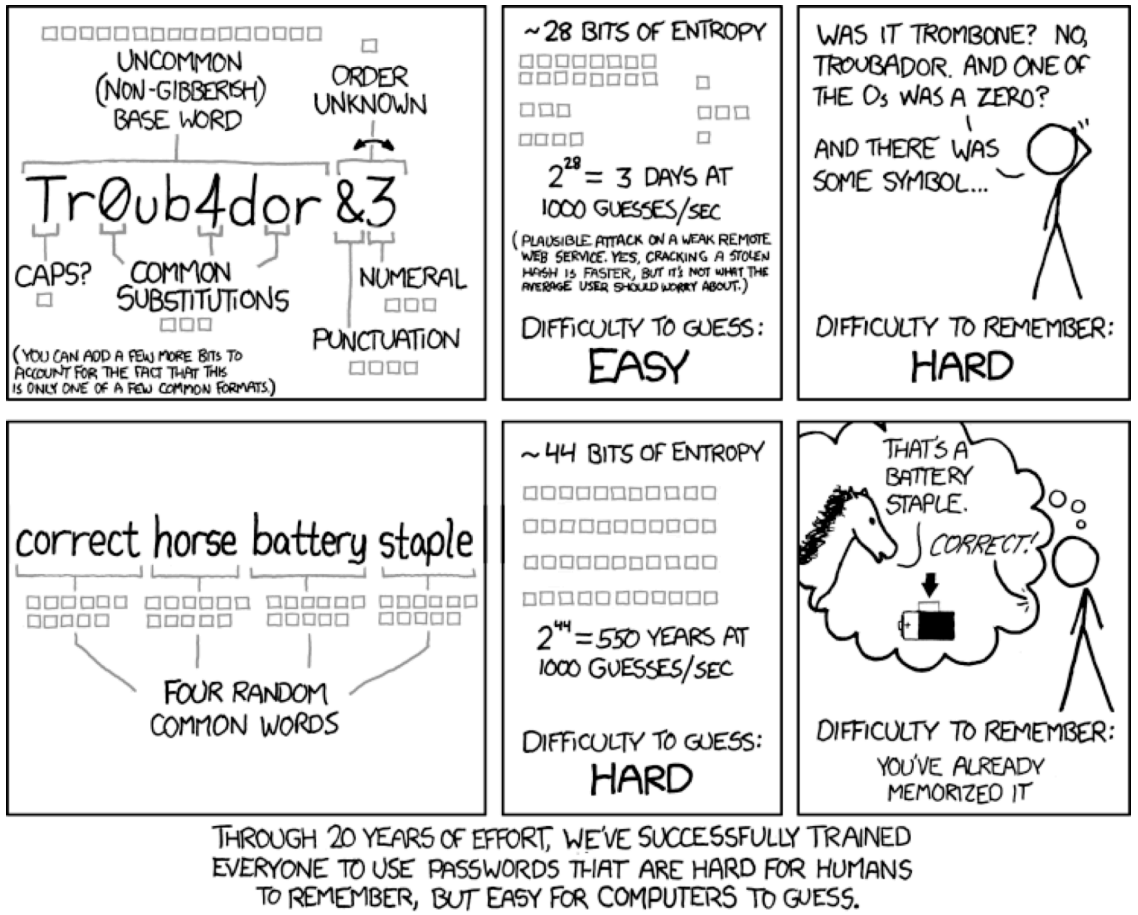
- **How Secure Is My Password?**
  https://howsecureismypassword.net

- **Try some passwords:**
  password
  password123
  montana2019
  Tr0ub4dour&3
  correcthorsebatterystaple
  correct horse battery staple

- Characteristics of ***strong*** passwords?

- Characteristics of ***weak*** passwords?

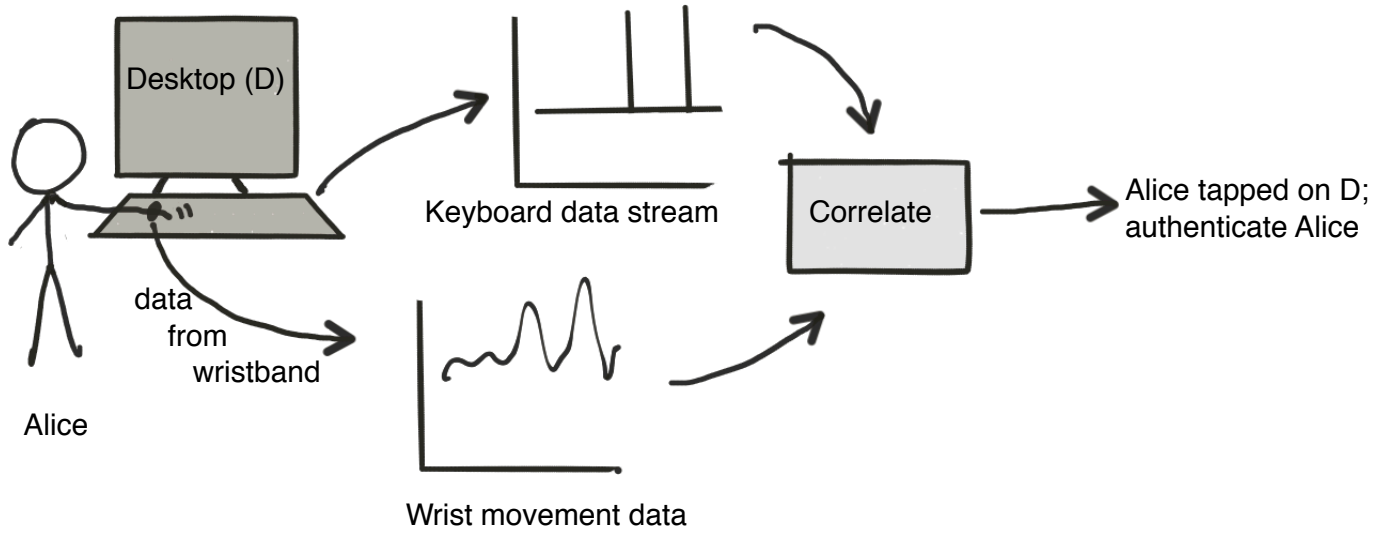
https://www.xkcd.com/936/



The Future of Authentication?

Idea: Secure, Intentional Authentication

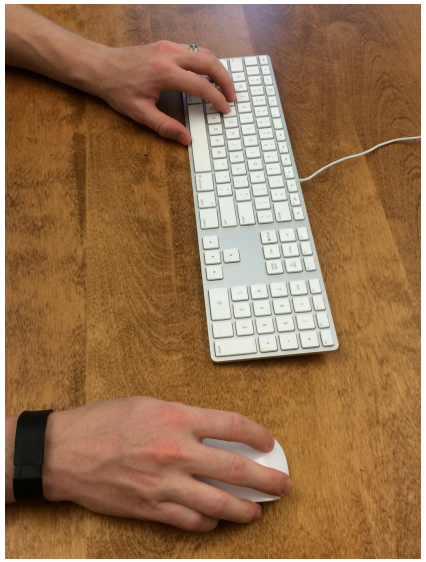
Shrirang Mare et al.
prof @ WWU, postdoc @ UW, phd @ Dartmouth



- SAW: Seamless Authentication with Wristbands – Initial Authentication
- CSAW: Continuous Authentication with Wristbands –Continuous Authentication

Idea: Secure, Intentional Authentication

CSAW: Continuous Authentication with Wristbands — **Continuous** Authentication



I_0, t_s^0, t_e^0

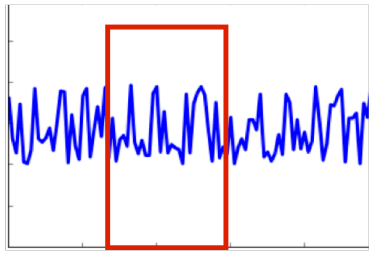
Actual

Sequence of Interactions

Interactions

1. Typing
2. Scrolling
3. Mouse-Keyboard switch

Correlate



t_s^0 t_e^0

F_0

Classifier

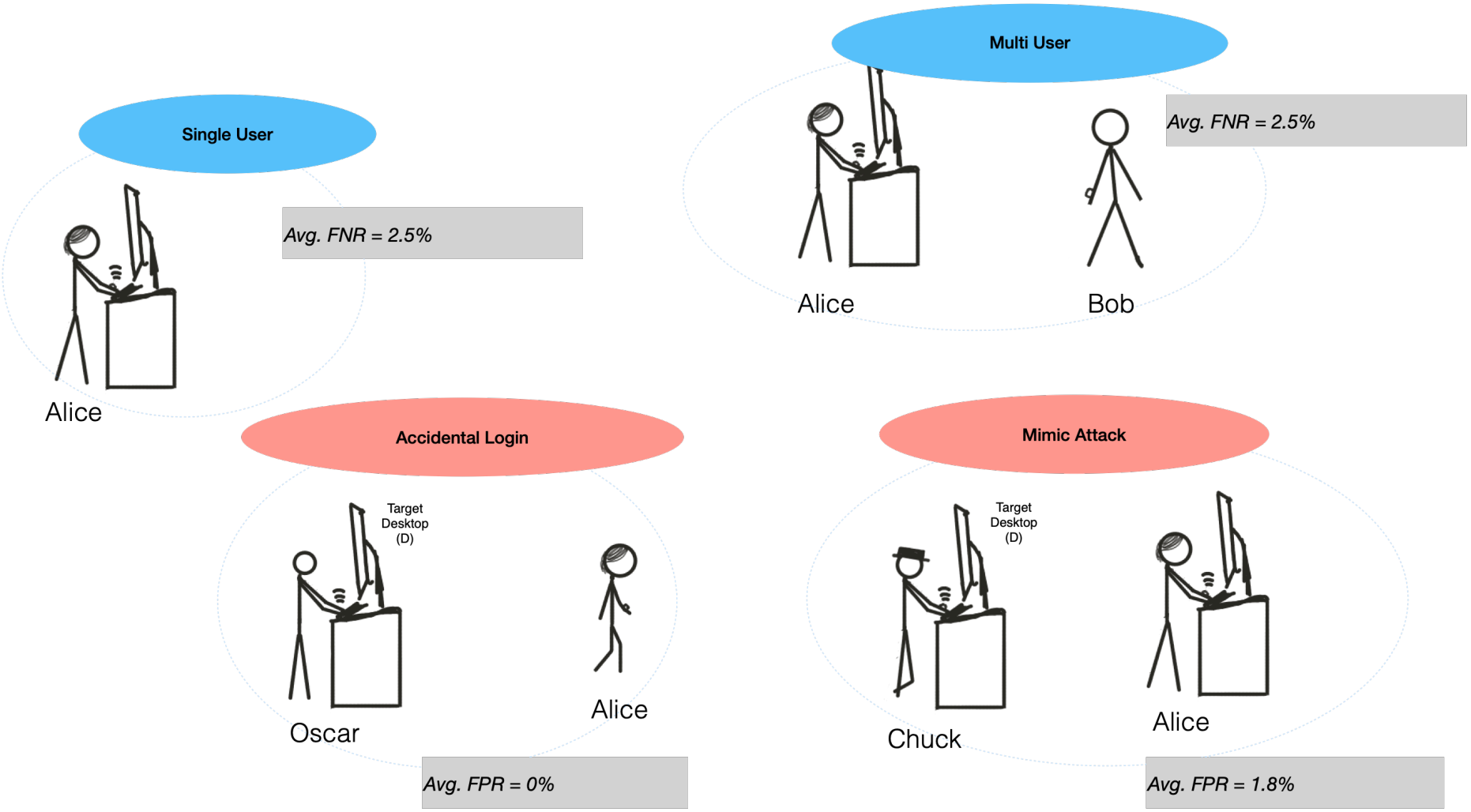
I_0', t_s^0, t_e^0

Sequence of Interactions

Estimated

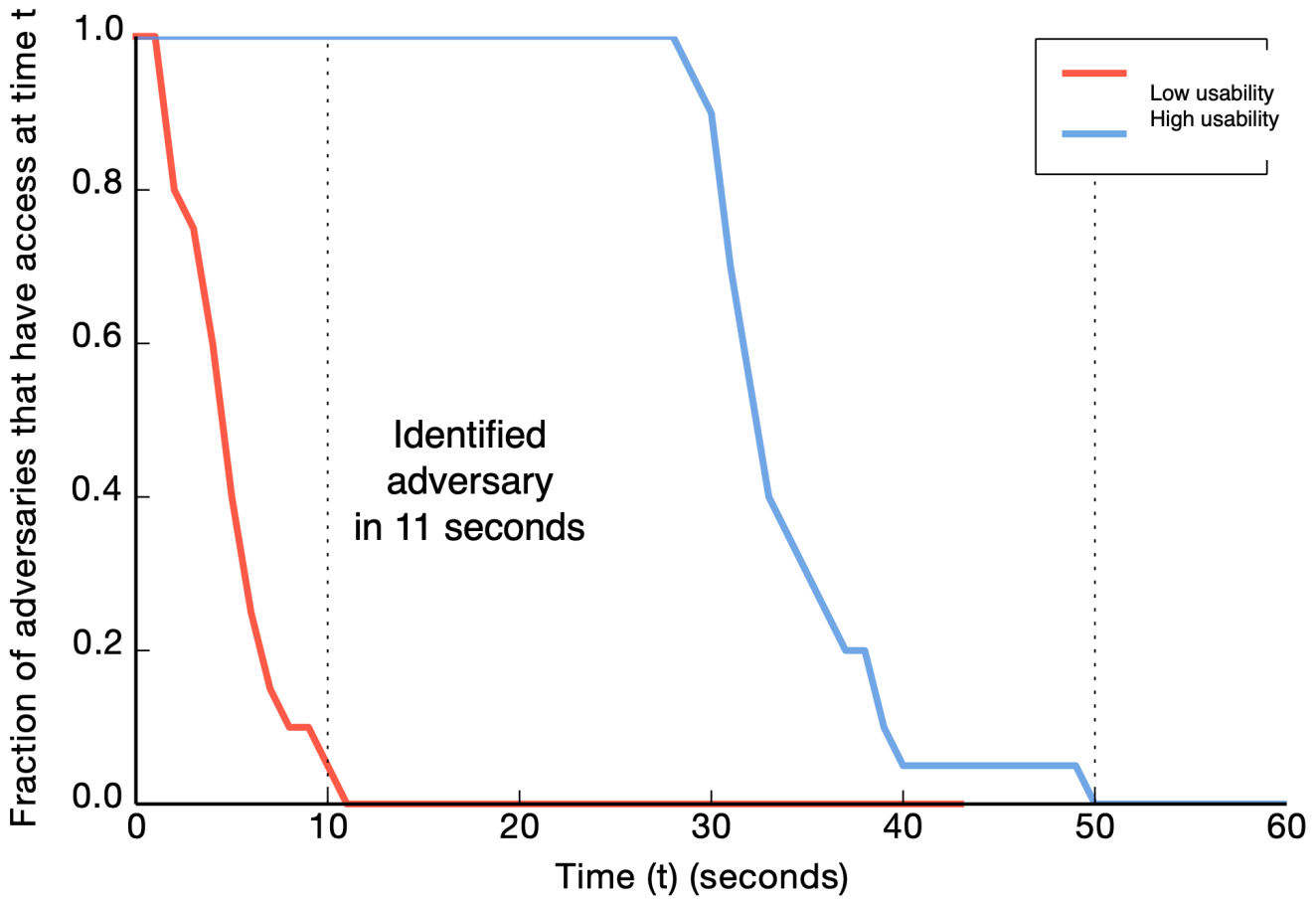
Idea: Secure, Intentional Authentication

SAW: Seamless Authentication with Wristbands – **Initial** Authentication



Idea: Secure, Intentional Authentication

CSAW: Continuous Authentication with Wristbands — **Continuous** Authentication



Fraction of **adversaries** that have access at time *t*

Trusted Execution Env. (SGX) & Trusted I/O

See slides from Travis's talk:

https://www.traviswpeters.com/slides/2018-hasp-bastionsgx-slides.pdf