OS Security

Professor Travis Peters
CSCI 460 Operating Systems
Fall 2019

Will Peteroy

Prep Resources (Read Before Class!)

Reading / Prep *(Read Before Class)*

- https://msrc-blog.microsoft.com/2010/12/08/on-the-effectiveness-of-dep-and-aslr/
- https://msrc-blog.microsoft.com/2013/08/12/mitigating-the-ldrhotpatchroutine-depaslr-bypass-with-ms13-063/
- https://arstechnica.com/information-technology/2019/08/armed-with-ios-0days-hackers-indiscriminately-infected-iphones-for-two-years/

Case Study 1 - Effectiveness of DEP and ASLR / Bypasses

- https://msrc-blog.microsoft.com/2010/12/08/on-the-effectiveness-of-dep-and-aslr/
- https://msrc-blog.microsoft.com/2013/08/12/mitigating-the-ldrhotpatchroutine-depaslr-bypass-with-ms13-063/

Case Study 2 - Apple - Extreme 0days in 2019

- https://arstechnica.com/information-technology/2019/08/armed-with-ios-0days-hackers-indiscriminately-infected-iphones-for-two-years/
- https://googleprojectzero.blogspot.com/2019/08/a-very-deep-dive-into-ios-exploit.html

A Guest Lecture w/

Will Peteroy — CEO @ icebrg.io

11/04/2019

https://www.traviswpeters.com/cs460/notes/cs460-22-os-security-will-peteroy.pdf

A Guest Lecture w/

Daniel Pagan — OS Enthusiast & Pen Tester @ IBM X-Force Red

11/13/2019

https://www.traviswpeters.com/cs460/notes/cs460-23-os-security-daniel-pagan.pdf

Tying Up Some Loose Ends in OS

Professor Travis Peters
CSCI 460 Operating Systems
Fall 2019

Threats

- Intruders
  - Masquerader *Non-authorized actor that tries to impersonate a legitimate user / exploit a legit. user's access privileges (**outsider**)*
  - Misfeasor *A legit. user that abuses their inside access to carry out nefarious deeds (**insider**)*
  - Clandestine User *Any entity that seizes privileged control of a system and evades detection*

- Malicious Software ("Malware")
  - Trojan horse *code that misuses its environment*
    - classic examples: NetBus, Back Orifice (BO), Back Orifice 2000 (B02K)
  - Backdoor *specific user identifier or password that bypasses normal security procedures*
    - classic examples: LiteBot, Remote Connection (RedNeck)
  - Stack and Buffer Overflow *exploits overflow bug to obtain ability to execute arbitrary code*
    - classic examples: Internet worm (fingerd), Code Red (IIS), SQLSlammer (MS SQL Server)
  - Spyware *SW that installs itself on a computer, reports personal info or activities*
    - classic examples: adware, stealware

Security Goal

Prevent attempts to gain unauthorized privileges

(or at least detect it…)

General Countermeasures / Defenses

(Multi-Factor) Authentication

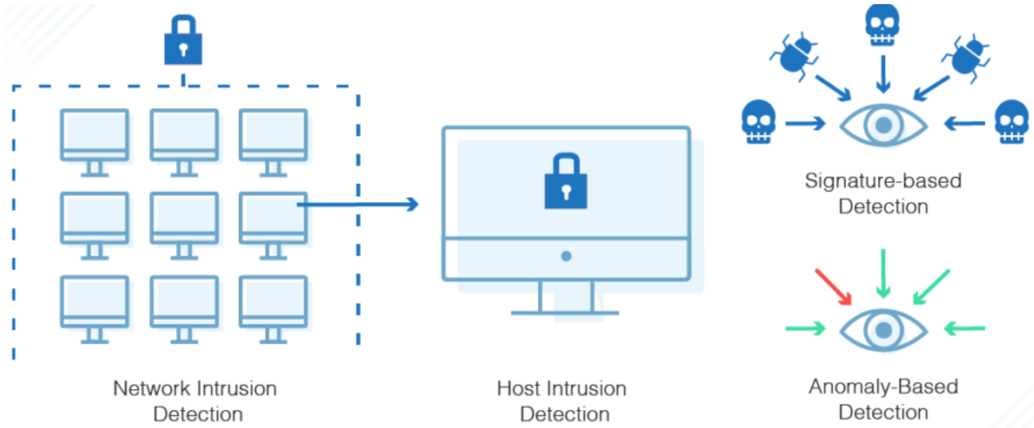
Identification

vs.

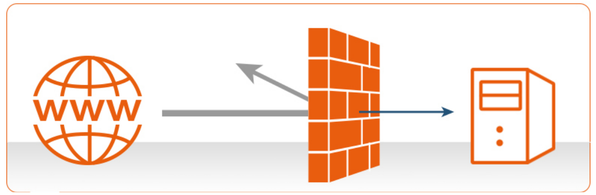
Verification

& Access Control *(later)*

Intrusion Detection Systems (IDS)



Firewalls



"Hardening" the OS/system

- Update and install patches
- Remove unnecessary services, applications, and protocols
- Configure appropriate users, groups, and permissions + resource controls
- Install additional security tools/controls
- Test
- *(Repeat)*

Exploitation Example: The Buffer Overflow

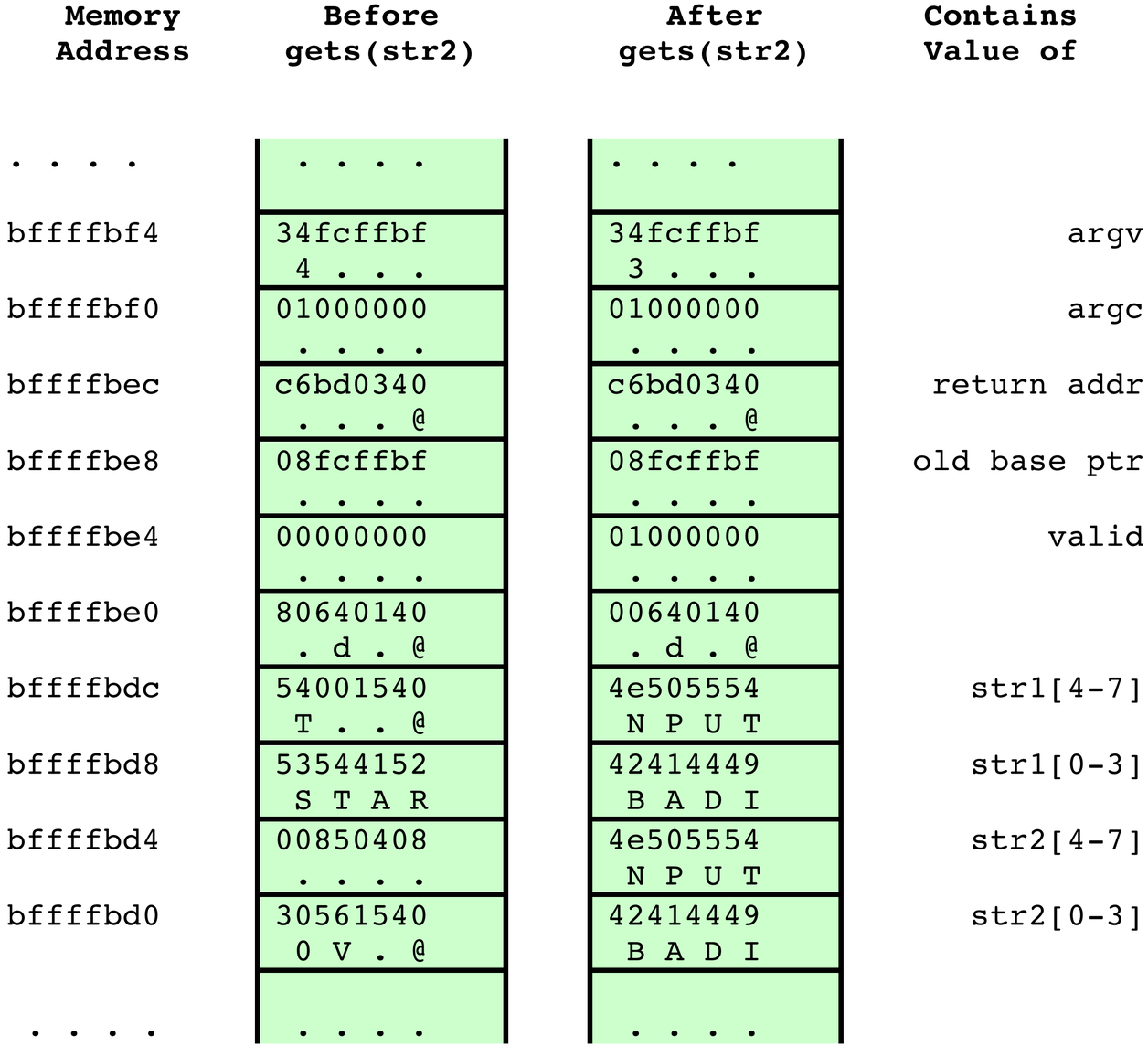
```
int main(int argc, char *argv[]) {
    int valid = FALSE;
    char str1[8];
    char str2[8];

    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

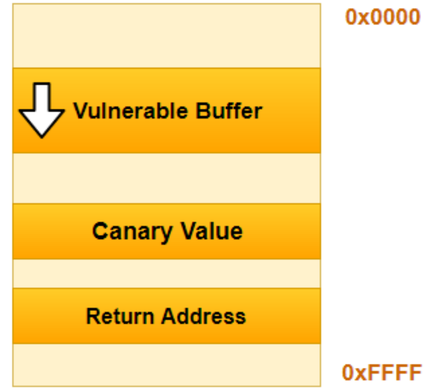
```
$ cc -g -o buffer1 buffer1.c
$ ./buffer1
START
buffer1: str1(START), str2(START), valid(1)
$ ./buffer1
EVILINPUTVALUE
buffer1: str1(TVALUE), str2(EVILINPUTVALUE), valid(0)
$ ./buffer1
BADINPUTBADINPUT
buffer1: str1(BADINPUT), str2(BADINPUTBADINPUT), valid(1)
```

(b) Basic buffer overflow example runs

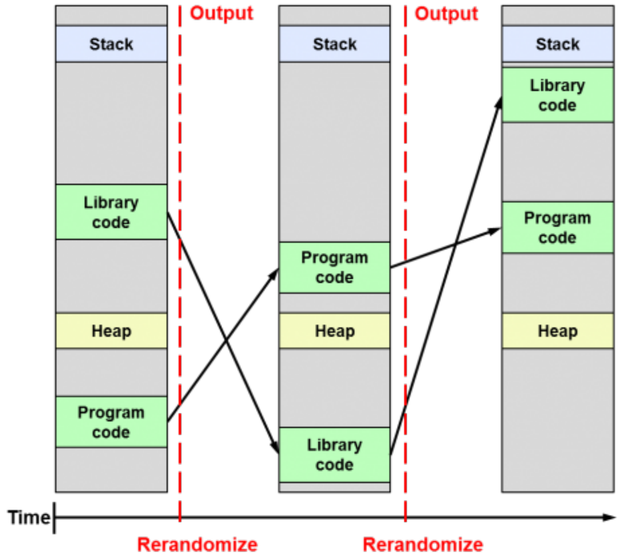


Overflow Countermeasures / Defenses

- Compile-Time Defenses
  - High-Level Programming Languages
  - Scrutinize code *open source vs. closed source…*
  - Language Extensions *use safe, validated implementations of libraries libsafe*
  - Stack Protection *stackguard (gcc); "canaries" → favor aborting/crashing programs over being pwned*



- Run-Time Defenses
  - Executable Address Space Protection *nx-bit on x86*
  - ASLR
  - Guard Pages



Access Control

*For each protected resource, store a list of **who** is permitted to do **what***

Access Matrix

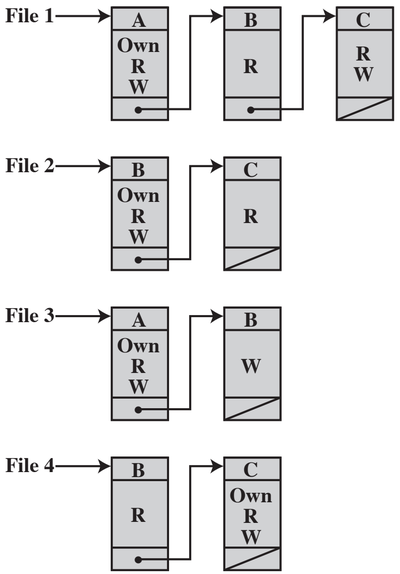
		OBJECTS			
		File 1	File 2	File 3	File 4
SUBJECTS	User A	Own Read Write		Own Read Write	
	User B	Read	Own Read Write	Write	Read
	User C	Read Write	Read		Own Read Write

For each subject → easy to look up their capabilities

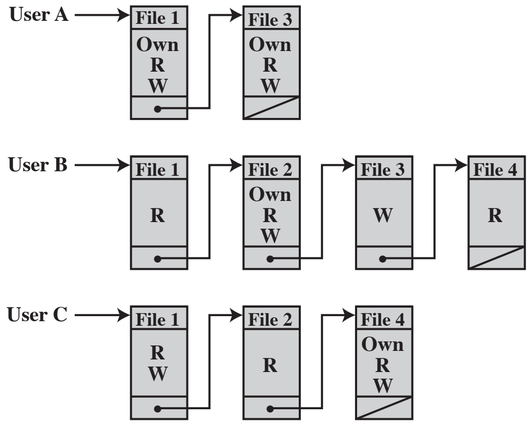
For each object → easy to look up authorized users

Access Control List (ACL)

objects → subjects

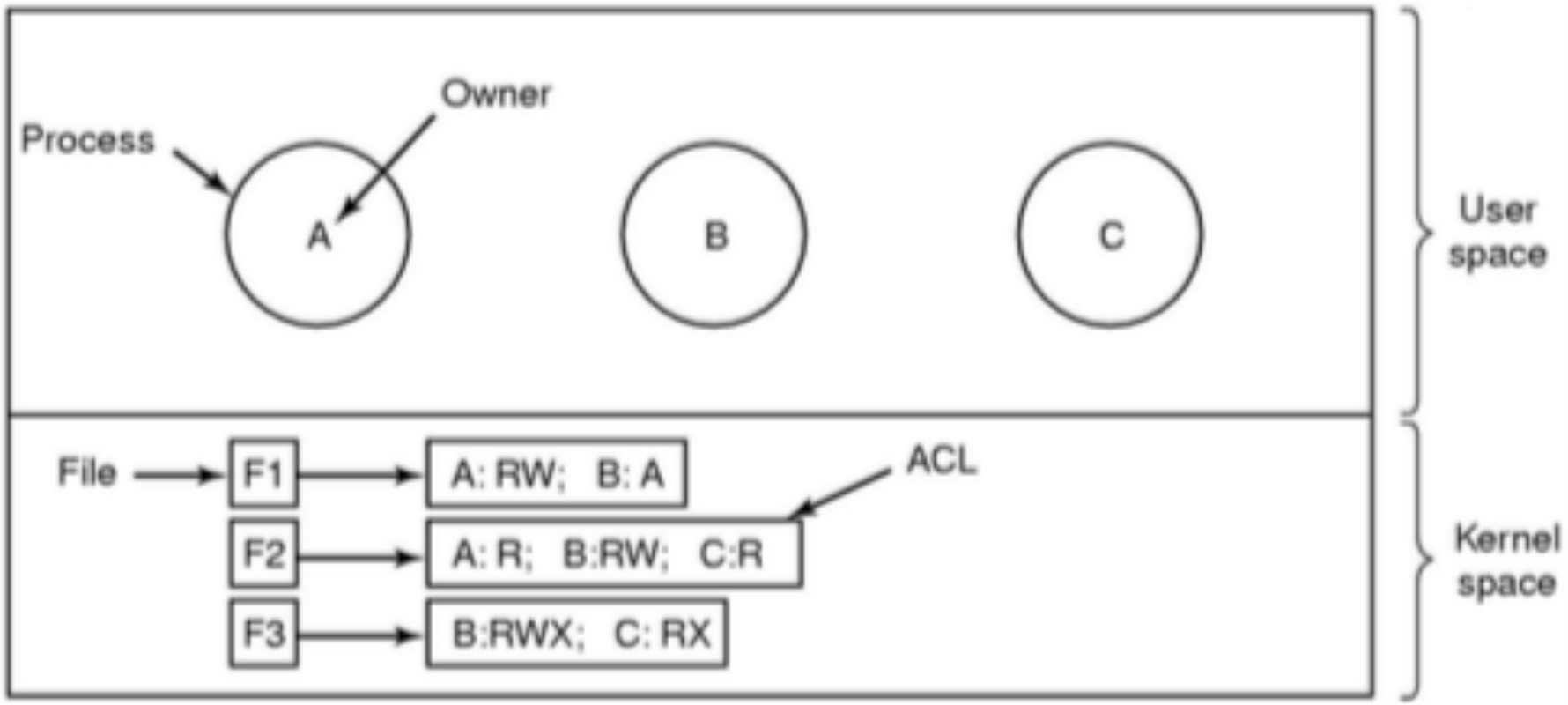


subjects → objects



Access Control *(cont.)*

Example: Use of access control lists to manage file access in UNIX



Access Control *(cont.)*

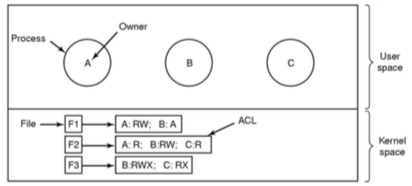
DAC

Each subject can perform action(s) on objects

subjects & objects each have associated security attributes

Permissions based on user/group

Discretionary *→ users can grant others access similar to its own permissions*



`Subject` can `Action` to `Object`

`Subject` can `grant` other `Subject`

MAC

Each subject can perform action(s) on objects

subjects & objects each have associated security attributes

Permissions based on user/group

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)*

`Subject` can `Action` to `Object`

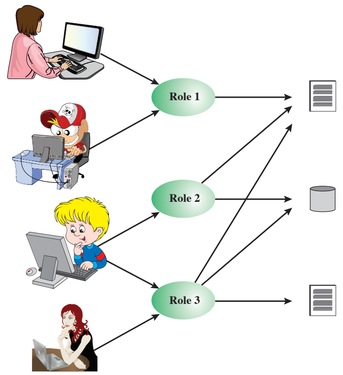
`Object` can be `Action` by `Subject`

RBAC

Roles have useful meaning and context for access permissions

Example: student, lab manager, professor, dean, president

Each ***subject*** *is mapped to a* ***role(s);*** *each role has* ***permission*** *to perform* ***actions*** *on* ***objects.***



`Subject` is a `Role`
which has `Permission` of `Action` to `Object`

ABAC

Policies expressed as a complex Boolean rule set that can evaluate many different attributes

Examples:

Subject Attributes
(age, clearance, dept. role, job title)

Action Attributes
(read, write, view, approve)

Resource Attributes
(object type, sensitivity, location)

Contextual Attributes
(time, location)

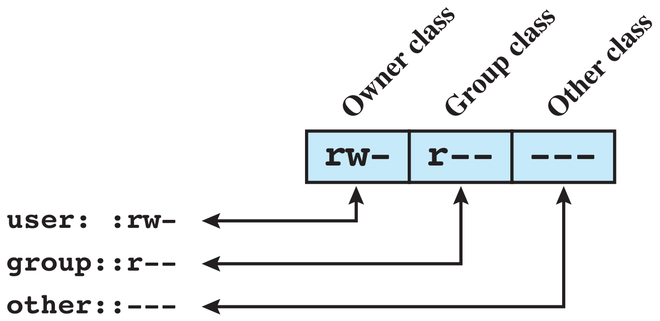
`Subject` who is xxx can `Action` to `Object` which is xxx in `Environment`

Some examples drawn from: https://dinolai.com/notes/others/authorization-models-acl-dac-mac-rbac-abac.html

Some Examples of Access Control IRL

Basic UNIX File Access Control

Every file has an owner and group owner.
Define r/w/x permissions for owner, group members, and everyone else (other)



`setuid` — *program runs with permission of principle who installed it / owns it.*

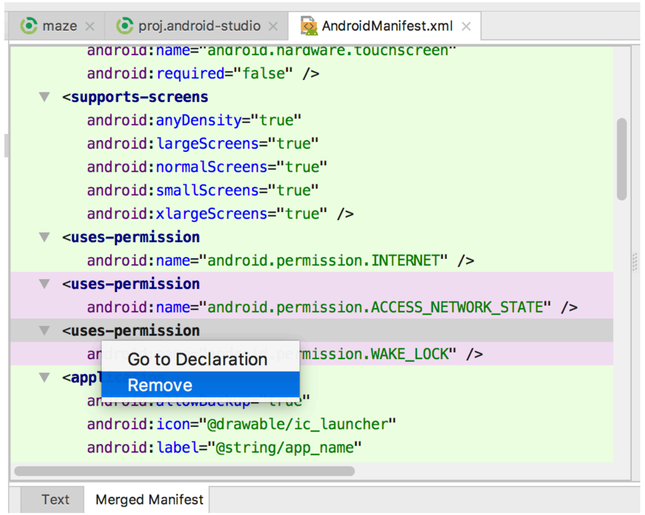
→ *if owner == superuser, and* `setuid` *bit is set, then program executes with superuser privileges!*

(Try it!) *Use* `stat` *to look at file details*

Smartphones & App Permissions

There exists a mechanism for apps to declare needed permissions and request permissions.

E.g., `AndroidManifest.xml`



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