

SECURITY

LECTURE 11 / FIT2100 / SEMESTER 2
2019

WEEK 12



SECURITY

INTRODUCTION

LEARNING OUTCOMES

- Describe the two main categories of system threats.
- Identify examples of how OS components add protection against security threats.
- Explore the limitations of security on typical operating systems
- Discuss how the user's security and privacy needs may be jeopardised by an operating system in a state of decay.

FURTHER READING

- Stallings chapter 16
- darkpatterns.org

SYSTEM THREATS

...FALL INTO TWO MAIN CATEGORIES

INTRUDERS

- Unauthorised access to a system by a user

MALICIOUS SOFTWARE

- Viruses and other forms of malware which may be present on the system.

INTRUDERS



INTRUDERS

EXTERNAL ATTACKERS

- An intruder may be a person attempting to access a system
 - Not authorised to do so
 - May even be a software bot, simulating a user

AUTHENTICATION

- A system of verifying that the user is authorised to access part of a system
- Linux systems: **usernames** and **passwords** guard access to the system.
- The username is **public** information.
- The password is a **secret** known only to the authorised user.

HOW DOES THE FILESYSTEM HELP? (1/2)

AUTHENTICATION

EACH FILE IN A UNIX FILESYSTEM CONTAINS A USER ID (UID)

- Each **user** on the system has a unique UID number.
- UNIX filesystems attach a UID (User ID) number to each **file**, and a set of permissions
 - Also a GID (group ID) for files shared among multiple users.
- A UID is also attached to each **process** on the system
 - Stored in the process control block
 - When a process is forked from another, the child process inherits the same UID: it belongs to the same user as the parent.

HOW DOES THE FILESYSTEM HELP? (2/2)

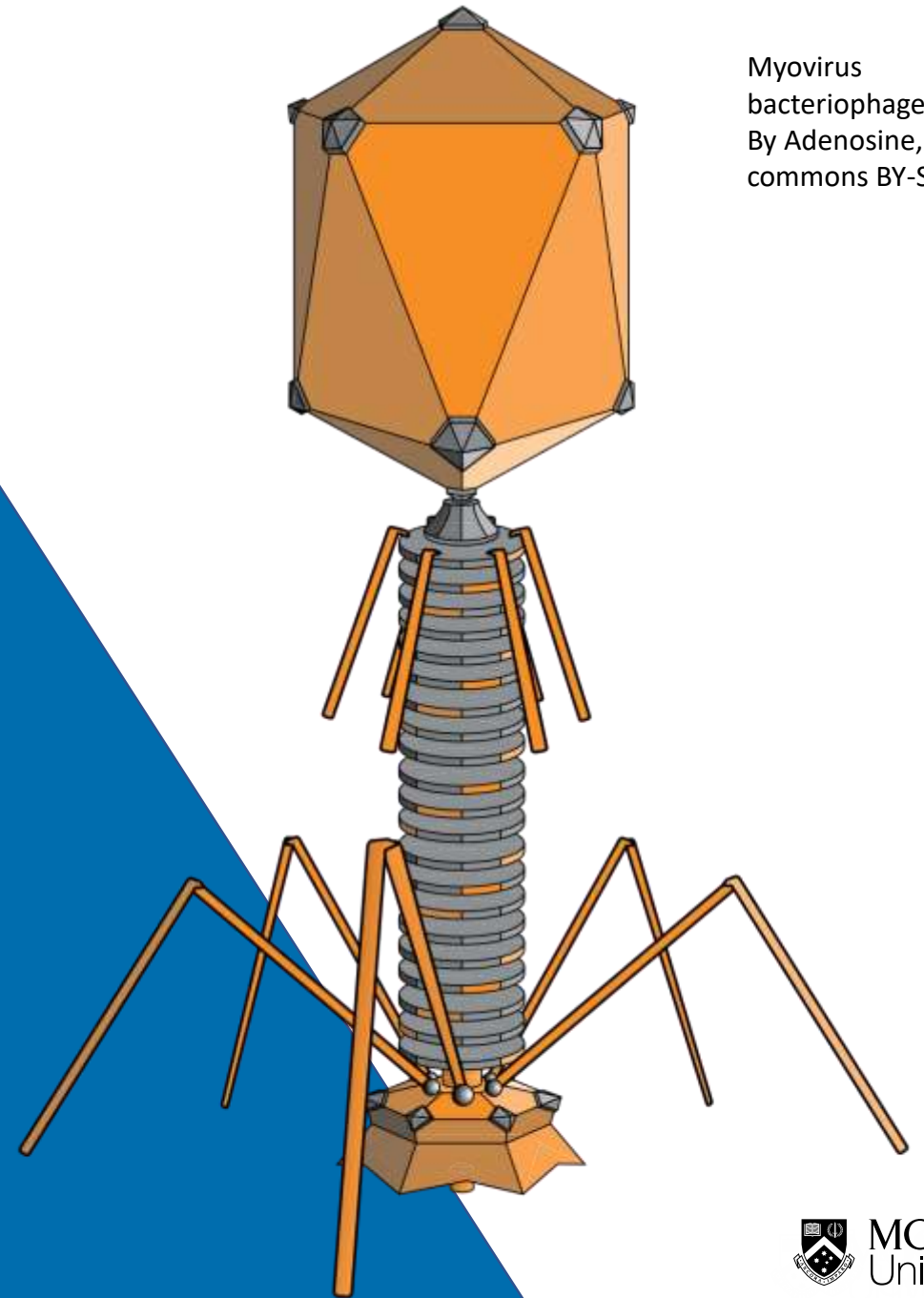
AUTHENTICATION

- When a process tries to **open()** a file, the kernel checks the **process's UID** against the **file's UID** and permissions
 - If the file and process have the same owner, or file permissions allow, the system call is allowed to complete. Otherwise the OS kernel rejects it.

DESIGN FLAWS

- Many filesystems (e.g. FAT) do not support UIDs
- When moving a hard drive or USB stick between systems, the UIDs will be different.
 - The file will appear to belong to a different user on the other system!
 - An authentication system designed a long time ago when all users were logged into the **same** computer system, maintained behind closed doors.
 - In the 1970s, the idea of moving a hard drive to another machine was unthinkable!

MALICIOUS SOFTWARE



Myovirus
bacteriophage.
By Adenosine, Creative
commons BY-SA 3.0

MALWARE

SHORT FOR 'MALICIOUS SOFTWARE'

TYPES OF MALWARE

1. Parasitic (viruses)

- Cannot exist outside some application, utility or system program
- e.g. a fragment of malicious code that has become embedded within a document or legitimate application.

2. Independent (worms, bots, trojans)

- A malicious utility that can be scheduled and run by the OS

3. Replicating

- Viruses or programs that produce copies of themselves.

HOW DOES VIRTUAL MEMORY HELP?

MEMORY PROTECTION EXAMPLES

A VIRUS MAY ATTEMPT TO INJECT ARBITRARY CODE INTO A PROCESS IMAGE

Pages or segments containing executable program code are read-only and cannot be modified by the running process.

- The virtual memory system marks entries in the page/segment table as read-only if they are allocated to executable machine code.
- Even if a program is compromised by loading malicious data, the program cannot re-write its executable own instructions.

COULD A MALICIOUS PROGRAM ATTEMPT TO SNOOP ON OTHER PROCESSES?

For example, could a malicious process attempt to read a list of passwords stored in another process's variables? By scanning different memory addresses?

- The virtual memory system ensures that the logical address space of one process does not permit access to the logical address space of another, unless both processes deliberately use shared memory.

WHAT ABOUT IPC? (1/3)

INTERPROCESS COMMUNICATION: A BIG CONCERN

A CLIENT PROCESS COULD EXPLOIT A VULNERABILITY IN A SERVER PROCESS

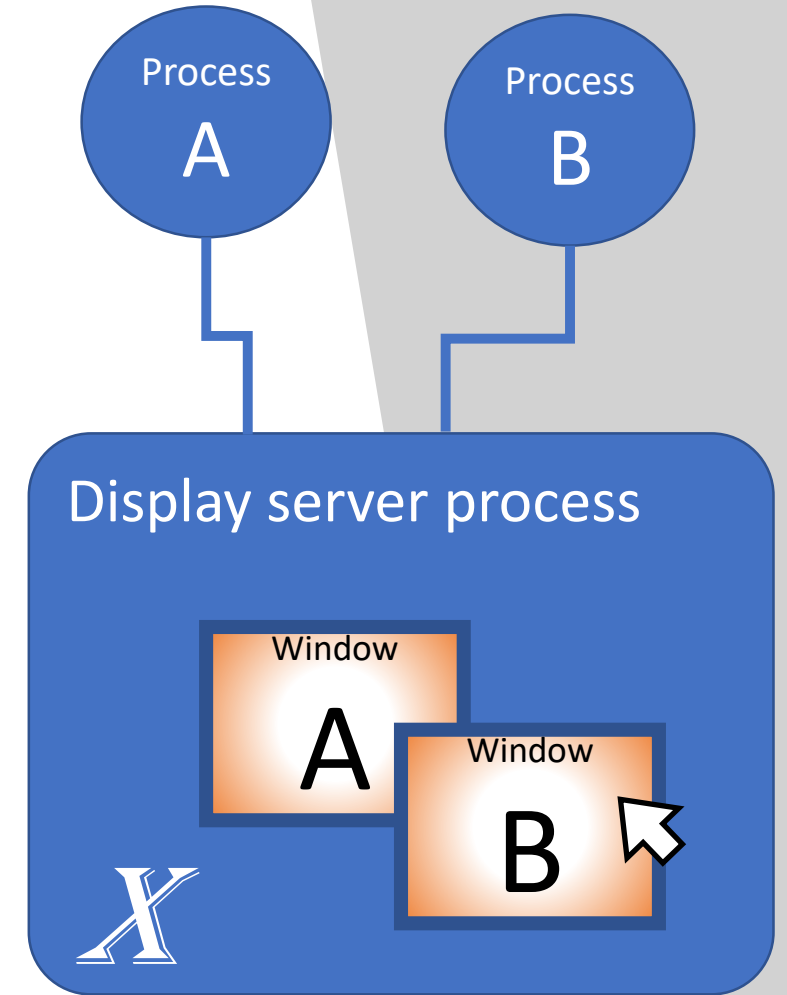
- **Buffer overflow attack**
 - A client process connects to a server process, then floods it with invalid data designed to exploit bugs in the server
 - The data sent goes past the end of the receiving buffer and overwrites other variables in the server; modifying the server's behaviour.
- **Unauthorised clients**
 - A client process might not be what it claims to be. A server process that is written to 'trust' a connected client may open up a security hole.
 - Server process might be running as a different user, giving a client access to a different user's files.

WHAT ABOUT IPC? (2/3)

A REAL-LIFE EXAMPLE: THE NATIVE DESKTOP

THE X-WINDOW SYSTEM IN LINUX

- The traditionally-used display server utility that allows client applications to place windows on the screen
 - Used in many Linux desktop environments
 - Including your virtual machine environment
- Each window on the screen is allocated a unique **window ID** (wid).
- To read or modify text or graphics in its window on the screen, a client application passes its **wid** to the X display server.
 - The X-Window system does not check *which* process is requesting access to a particular **wid**
 - **Any** process can access any other process's window contents freely.
 - Similar issues traditionally existed in Windows.



WHAT ABOUT IPC? (3/3)

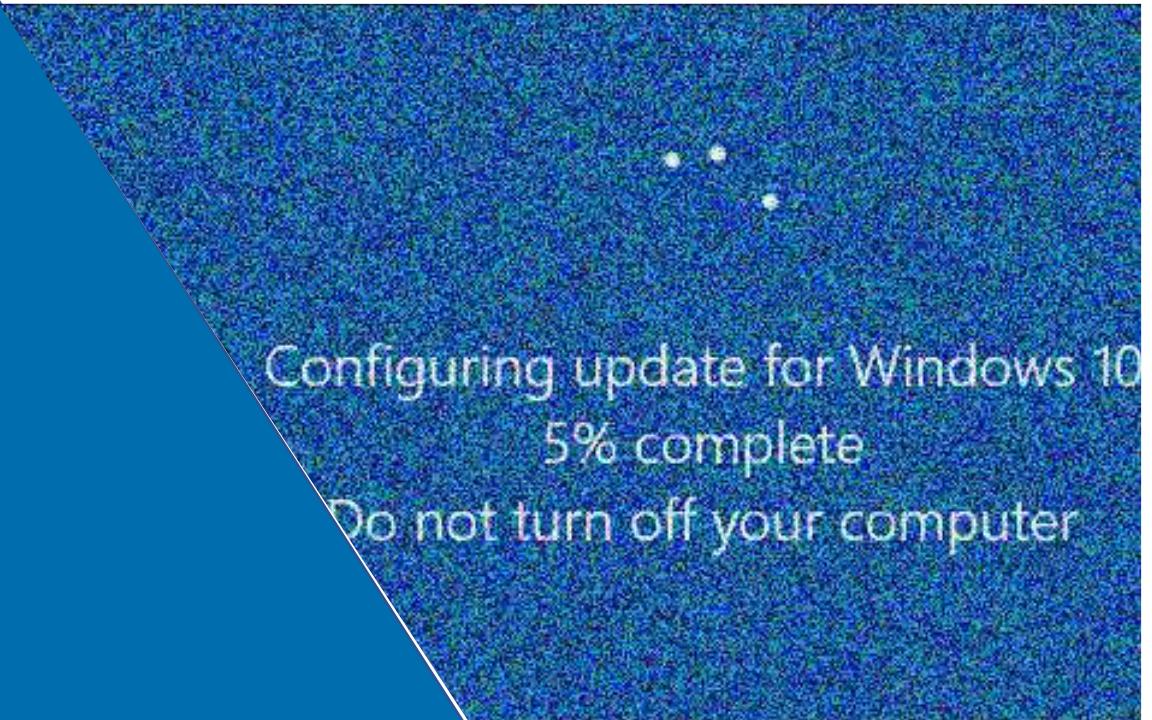
WHAT CAN WE LEARN FROM THIS?

- Even if the OS is secure by itself, insecure utilities can open access paths to unauthorised users and malware.
- Usually by accepting connections through IPC mechanisms.

WHAT CAN BE DONE ABOUT THIS?

- Programs that accept client connections may be run under a special user account
 - Does not give access to sensitive files on the system
- The best fix is to write utilities carefully so they are not vulnerable to security threats
 - Authenticate new connections, validate incoming data.

OPERATING SYSTEMS IN DECAY



Configuring update for Windows 10
5% complete
Do not turn off your computer

THE CASE OF A DYING OPERATING SYSTEM

ALL PRODUCTS HAVE A LIMITED LIFESPAN

MAINTENANCE FACTORS

- Design rot
 - Over time, an operating system's code becomes more difficult to maintain as the code itself moves away from the original design.

when fixing or patching a bug, the code will move away from original whole purpose of the OS, this may cause OS to decline with time

BUSINESS FACTORS

- Dark patterns
 - Over time, the operating system vendor's priorities may shift away from designing the OS around goals such as **efficiency**, **privacy** and **usability**, towards meeting goals that maximise the OS vendor's revenue rather than the needs of users.

they stop caring about the technical aspects but more on its revenue to keep the company going, causing the OS performance to decline

DESIGN ROT

NEW SOFTWARE TENDS TO BE BUILT AROUND A CLEAR DESIGN FRAMEWORK

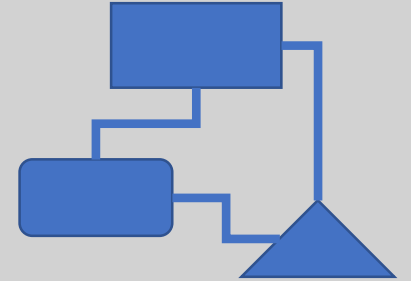
ANY PIECE OF SOFTWARE BECOMES MORE COMPLEX OVER TIME

- As software is maintained, faults in the system often require fixing
- If the problem is due to the design of the operating system, it might not be feasible to **re-design** the OS so the fault no longer exists.
 - Redesigning a system may create **new** faults and break compatibility for users.
- Quick fixes and 'patches' make the original design of the system less clear.

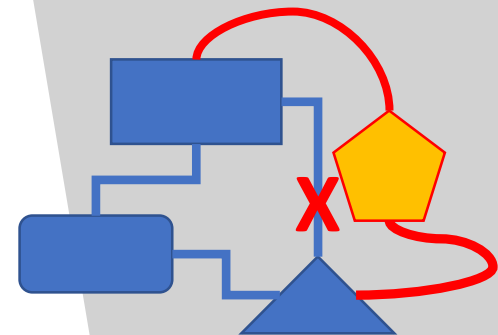
TOO MUCH COMPLEXITY MAKES SECURITY IMPOSSIBLE

- Updates intended to fix security issues usually make the operating system **more complex**.
- As the OS becomes **too complex**, the workings of the **OS itself** can no longer be easily scrutinised.
- Eventually, the system is no longer trusted by users.

ORIGINAL
CLEAR
DESIGN



PATCH TO FIX
PROBLEM



DARK PATTERNS

DARK PATTERNS

TERM COINED BY HARRY BRIGNULL, 2010, TO DESCRIBE USER INTERFACE TRAPS FOUND ONLINE

When the design of the system becomes focused the vendor's sales goals rather than needed functionality.

Examples:

- Repeatedly interrupting the user to prompt them to carry out a task that does not benefit them
- Hiding access to desirable options

You haven't set up Skype yet!

Set up Skype now to start talking to your friends for FREE*

Let's GO!!

Remind me later...

Remind me NOW

Unsolicited notifications like these are not relevant to the task the user needs to achieve; usually designed to manipulate the user into agreeing to a set of terms and conditions.

OTHER NEGATIVE DESIGN PATTERNS

BUILT-IN OBSOLESCENCE

Changes are implemented for the sake of making changes

Less focus on innovation, more focus on 'keeping up appearances'

A system may be changed to break compatibility with existing software

Users may be left with no alternative but to change to a newer system

SUMMARY (LECTURE 11)

SECURITY

- The two main categories of system threats are **intruders** and **malware**.
 - The filesystem and virtual memory system both provide degrees of protection against unauthorised access
 - Even with security measures built into the OS itself, an insecure server utility may open up a path of access for an intruder or malicious software client.
 - When an operating system begins to exhibit **design rot** and **dark patterns**, users may no longer be able to verify, or have confidence, that the OS is secure.
-
- **Next week: Swot vac**