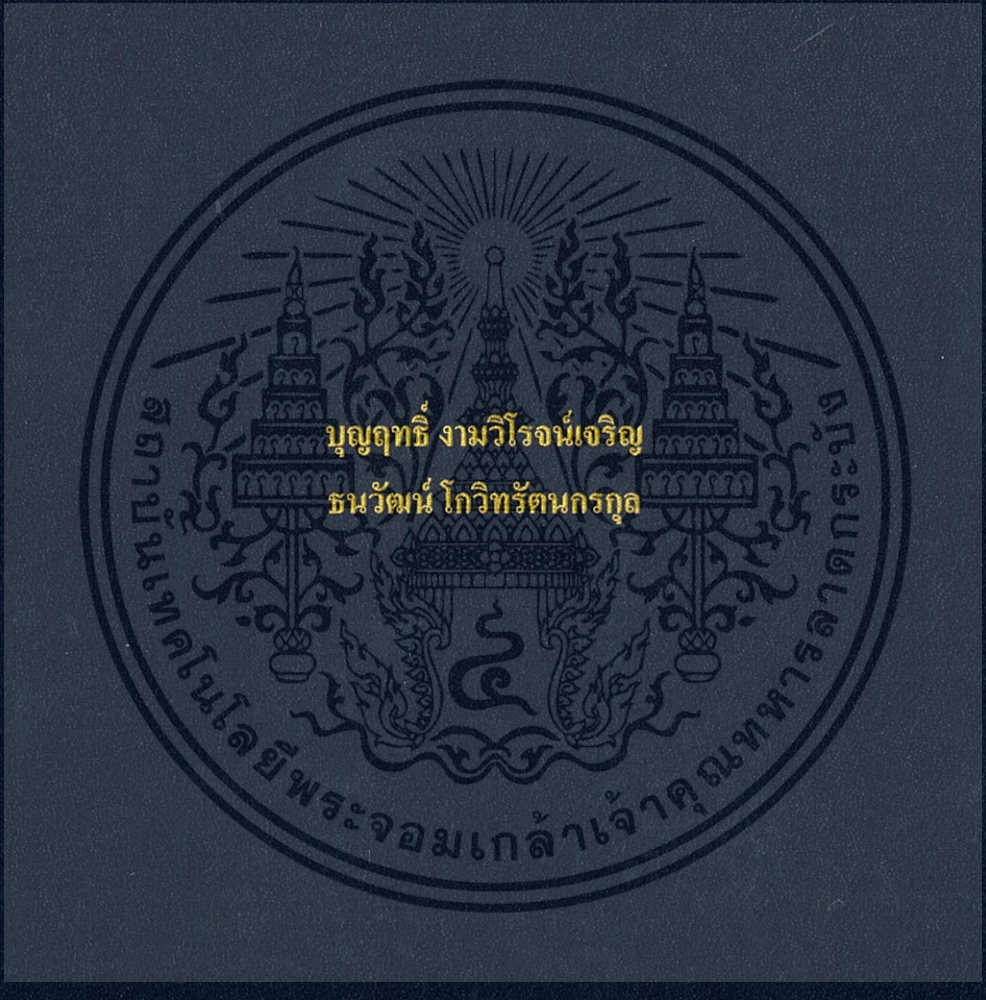
ระบบฝึกหัดเพื่อการเขียนโค้ดให้ปลอดภัย

SECURE CODING TRAINING SYSTEM



บุญฤทธิ์ งามวิโรจน์เจริญ

ธนวัฒน์ โกวิทรัตนกรกุล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2558

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

ระบบฝึกหัดเพื่อการเขียนโค้ดให้ปลอดภัย

SECURE CODING TRAINING SYSTEM

บุญฤทธิ์ งามวิโรจน์เจริญ

ธนวัฒน์ โกวิทรัตนกรกุล

b.10819323
l.

เรขหมู่
เลขทะเบียน 144431
วัน.เดือน.ปี 24 พ.ย. 2559

รายงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2558

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รายงานปีการศึกษา 2558

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบฝึกหัดเพื่อการเขียนโค้ดให้ปลอดภัย

SECURE CODING TRAINING SYSTEM

ผู้จัดทำ

1. นายบุญฤทธิ์ งามวิโรจน์เจริญ รหัสนักศึกษา 55010683

2. นายธนวัฒน์ โกวิทรัตนกรกุล รหัสนักศึกษา 55010511

_____อาจารย์ที่ปรึกษา

(อาจารย์อัครเดช วัชระภูพงษ์)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ระบบฝึกหัดเพื่อการเขียนโค้ดให้ปลอดภัย

นายบุญฤทธิ์	งามวิโรจน์เจริญ	55010683
นายธนวัฒน์	โกวิทรัตนกรกุล	55010511
อาจารย์อัครเดช	วัชระภูพงษ์	อาจารย์ที่ปรึกษา

ปีการศึกษา 2558

บทคัดย่อ

โครงการจัดทำขึ้นเพื่อศึกษาช่องโหว่ที่เกิดขึ้นในการเขียนภาษา C และ C++ ต่างๆ เช่น Interger Overflow , การเกิด Buffer overflow ที่เกิดขึ้นใน formatted output และการเกิด Unbounded String Copies เป็นต้น อย่างไรก็ตาม ช่องโหว่ที่กล่าวมาเหล่านี้อาจจะนำมาใช้เพื่อที่จะลักลอบเจาะมาที่เป้าหมายได้หรืออาจทำให้โปรแกรมทำงานผิดพลาดได้นั้นเอง ด้วยเหตุผลนี้โปรแกรมต่างๆที่เขียนขึ้นมาโดยปราศจากการคำนึงถึงเรื่องความปลอดภัยที่จะส่งผลร้ายตามมาต่อทั้งโปรแกรมของตนเองและระบบปฏิบัติการที่ใช้จึงเกิดขึ้นได้เสมอและปริญญานิพนธ์ฉบับนี้ศึกษาวิธีการเขียนโปรแกรมที่ดี เพื่อไม่ให้เกิดเหตุการณ์เหล่านั้นได้ โดยขั้นตอนการศึกษานั้นเราจะศึกษาตั้งแต่การโจมตี เพื่อเรียนรู้จุดอ่อนของระบบเพื่อที่จะเจาะเข้าไปได้ และเมื่อทราบถึงกระบวนการบุกรุกแล้ว ก็เป็นขั้นตอนการป้องกันเพื่อไม่ให้ก่อให้เหตุการณ์นั้นได้อีกในอนาคต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Secure Coding Training System

Mr. Boonyarit Ngamvirojcharoen 55010683
Mr. Thanawat Kowrattanakornkun 55010511
Mr. Akkradach Watcharapupong Advisor
Academic Year 2015

ABSTRACT

The project is designed to study vulnerability in languages C and C ++ such as Integer Overflow ,Buffer Overflow in formatted output and Unbounded String Copies etc. All of this vulnerability may be the cause to attack from hacker . For this reason some programs that programmer not concerns for security in his program will make some damage to his program either his . This thesis concerns with the study of how to programming secure program to prevent the attack from hacker. First section for this thesis will explain how to attack and situation.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยดีด้วยคำแนะนำและคำปรึกษาจาก อาจารย์อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษา ขอขอบคุณอาจารย์คณะวิศวกรรมศาสตร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ทุกท่านที่ประสิทธิ์ประสาทความรู้จนสามารถนำความรู้ไปประยุกต์ใช้ทำงานวิจัยได้ สุดท้ายนี้ทางคณะผู้จัดทำต้องขอบพระคุณเป็นอย่างสูง และ หวังเป็นอย่างยิ่งว่าปริญญานิพนธ์ฉบับนี้จะเป็นประโยชน์แก่ผู้อ่านไม่มากก็น้อย

บุญฤทธิ์ งามวิโรจน์เจริญ
ธนวัฒน์ โกวิทรัตนกรกุล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป(ต่อ)

รูป **หน้า**

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่1

บทนำ

1.1 ความสำคัญและที่มาของโครงงาน

เนื่องจากการเกิดช่องโหว่ในภาษา C และ C++ สามารถที่จะเกิดขึ้นได้ง่าย ยกตัวอย่างเช่น ในเรื่องของ String ช่องโหว่ที่มักเกิดขึ้น คือ Unbounded String จะเกิดขึ้นเมื่อมีการ copy ค่าจำนวนมากลงสู่ค่าของ array ที่มีการกำหนดค่าไว้ทำให้เกินค่าที่จองไว้ เป็นต้น ในเรื่องของ Pointer Subterfuge ช่องโหว่ที่มักเกิดขึ้น คือ Data Pointer Subterfuge ที่เป็นการเปลี่ยนจุดชี้ข้อมูล ทำให้โปรแกรมทำงานตามที่ต้องการของผู้โจมตีคล้ายกับการทำ Arc Injection เป็นต้น ในเรื่องของ integer ช่องโหว่ที่มักเกิดคือ การเกิด Integer Overflow จะเกิดขึ้นเมื่อมีการแทนค่าของ integer เกินกว่าค่าสูงสุดหรือการแทนค่าที่ต่ำกว่าค่าของ integer ซึ่งถ้าผู้ใช้ไม่มีความระมัดระวังก็อาจจะทำให้เกิด integer overflow ได้ เป็นต้น และ ในเรื่องของ Formatted Output ช่องโหว่ที่มักเกิดคือ เรื่อง Buffer overflow ในการใช้ formatted output เป็นต้น และในเรื่องของ File I/O ช่องโหว่ที่มักเกิดคือ ช่องโหว่ Symbolic Linking ซึ่งจะเกี่ยวข้องกับ TOCTOU เป็นต้น ซึ่งช่องโหว่ที่กล่าวมาเหล่านี้อาจจะนำมาใช้เพื่อที่จะลักลอบเจาะมาที่เป้าหมายได้หรืออาจทำให้โปรแกรมทำงานผิดพลาดได้นั้นเอง ด้วยเหตุผลนี้โปรแกรมต่างๆที่เขียนขึ้นมาโดยปราศจากการคำนึงถึงเรื่องความปลอดภัยที่จะส่งผลร้ายตามมาต่อทั้งโปรแกรมของตนเองและระบบปฏิบัติการที่ใช้จึงเกิดขึ้นได้เสมอ

ดังนั้นเราจึงควรศึกษาวิธีการเขียนโปรแกรมที่ดี เพื่อไม่ให้ก่อให้เกิดเหตุการณ์เหล่านั้นได้ โดยขั้นตอนการศึกษานั้นเราต้องศึกษาตั้งแต่การโจมตี เพื่อเรียนรู้จุดอ่อนของระบบเพื่อที่จะเจาะเข้าไปได้ และเมื่อทราบถึงกระบวนการบุกรุกแล้ว ก็จะเป็นขั้นตอนการป้องกันเพื่อไม่ให้ก่อให้เหตุการณ์นั้นได้อีกได้ในอนาคต

1.2 วัตถุประสงค์

1) เพื่อศึกษาช่องโหว่ของภาษา c การบุกรุกด้วยเทคนิคต่างๆ เช่น บัฟเฟอร์โอเวอร์โฟลด์
2) เพื่อศึกษาวิธีการป้องกันการบุกรุกช่องโหว่ด้วยวิธีการต่างๆ
3) หาวิธีเพื่อให้ผู้เขียนเขียนโปรแกรมตระหนักถึงการเขียนโค้ดให้ถูกต้องและมีช่องโหว่น้อยที่สุด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.3 ขอบเขตโครงงาน.

ภาษา c ด้าน secure coding

1.4 วิธีการดำเนินงาน

1) ศึกษารายละเอียดต่างๆ ของโปรเจค

-String Vulnerabilities

-Pointer Vulnerabilities

-Integer Vulnerabilities

- Formatted Output Vulnerabilities

- File I/O Vulnerabilities

2) ออกแบบระบบการสอนและสื่อการสอน

1.5 ประโยชน์ที่ได้รับ

1) ได้รับความรู้เกี่ยวกับช่องโหว่ทางภาษาซี
2) ได้รับความรู้เกี่ยวกับการป้องกันและหลีกเลี่ยงช่องโหว่ทางภาษาซี

1.6 ส่วนประกอบปริญญานิพนธ์

ปริญญานิพนธ์ประกอบด้วยบทจำนวน 5 บท

บทที่ 1 กล่าวถึงความสำคัญและที่มาของโครงงาน วัตถุประสงค์ของโครงงาน ขอบเขตของโครงงาน วิธีการดำเนินการ ประโยชน์ที่คาดว่าจะได้รับ และส่วนประกอบของปริญญานิพนธ์

บทที่ 2 กล่าวถึง ทฤษฎีที่เกี่ยวข้อง กล่าวถึง ทฤษฎีพื้นฐานที่ใช้ในการทำโครงงาน ประกอบด้วยช่องโหว่ต่างๆ ไม่ว่าจะเป็น buffer overflow, Stack Smashing, Code Injection, Arc Injection,Data Pointer Subterfuge, Integer Overflow ,Symbolic Linking และช่องโหว่อื่นๆ วิธีการหลีกเลี่ยงการเกิดช่องโหว่ไม่ให้เกิดขึ้นและการเขียนโปรแกรมที่ดี

บทที่ 3 กล่าวถึง ทฤษฎีการออกแบบและทฤษฎีการพัฒนาสื่อการสอน การนิยามของคำว่าการออกแบบระบบการสอน ประเภทสื่อการสอน

บทที่ 4 กล่าวถึง การทดลองและการประเมินผล การทดลองใช้เว็บไซต์ในส่วนต่างๆ และการประเมินผลการใช้งานเว็บไซต์

บทที่ 5 กล่าวถึง บทสรุป ปัญหา อุปสรรค และ แนวทางการพัฒนาต่อ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2

ทฤษฎีที่เกี่ยวข้อง

2.1 Common String Manipulation Errors

2.1.1 Unbounded String Copies

Unbounded String เกิดเมื่อมีการ copy ค่าจำนวนมากลงสู่ array ที่มีการ fix ขนาด

โปรแกรม 2.1 Reading unbounded stream from standard input

```
1.int main(void){
2. char Password[80];
3. puts("Enter 8 character password");
4  gets(Password);
 ...
```

จากภาพโปรแกรมอ่านค่าตัวอักษรจากอินพุตโดยใช้ฟังก์ชัน gets() ลงสู่ array ทีมีการกำหนดขนาดจนขึ้นตัวอักษรใหม่หรือจบไฟล์การอ่านข้อมูลจากข้อมูลจำนวนมาก สามารถสร้างปัญหาสำหรับโปรแกรมเมอร์ได้ เพราะว่าเราไม่มีทางรู้ล่วงหน้าได้ว่ามีจำนวนอักษรเท่าไรที่จะเราควรจัดสรรพื้นที่ให้มันซึ่งไม่มีทางเป็นไปได้ที่จะทำการจองพื้นที่ล่วงหน้าเอาไว้ได้ วิธีแก้ที่ง่ายจึงทำได้โดยการจองพื้นที่ล่วงหน้าด้วยการจองจำนวนตัวอักษรมาก ๆ จากตัวอย่างโปรแกรมเมอร์คาดว่าผู้ใช้จะใส่ข้อมูลไม่เกิน 80 ตัวอักษรแต่ถ้าผู้ใช้เกิดใส่เกิน จะทำให้เกิด error ขึ้นได้

ซึ่งมันสามารถสร้าง error ได้ง่าย ๆ เมื่อมีการใช้ ฟังก์ชัน copy และ ฟังก์ชั่น concatenating เพราะว่า การใช้ฟังก์ชัน strcpy() และ strcat() จะสามารถ copy อักษรได้ไม่จำกัด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรม 2.2 Unbounded string copy and concatenation

```
1.int main(int argc,char *argv[]){
2. char name[2048];
3. strcpy(name,argv[1]);
4  strcpy(name,"=");...
5 strcat(name,argv[2]);
  ...
```

จากโปรแกรม 2.2 เราสามารถ copy และ concat ได้เรื่อยๆ ไม่จำกัดจนเกิน 2048 ตัวอักษร

โปรแกรม 2.3 Dynamic allocation

```
1. int main(int argc,char *argv[]){
2.    char *buff =(char *)malloc (strlen(argv[1]+1));
3.    if (buff != NULL){
4     strcpy(buff,argv[1]);…
5 printf("argv[1] = %s.\n",buff);
6 }
7. else {
8. }
9. return 0;
10. }
```

วิธีแก้ปัญหาที่ง่ายคือการใช้ฟังก์ชัน strlen() และ การใช้ dynamically allocate เพื่อใช้ในการจัดการ memory

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.1.2 Off-by-One Errors

อีกปัญหาที่พบบ่อย กับ String ในภาษา C ที่เหมือนกับ unbounded string copies คือจะเกี่ยวข้องกับ การเขียนค่าใน array โดยจะเป็นการเขียนค่านอกขอบเขต ของ array

โปรแกรม 2.4 Common off-by-one defects

```
1. int main(int argc, char* argv[]) {
2.    char source[10];
3.    strcpy(source, "0123456789");
4.     char *dest = (char *)malloc(strlen(source));
5.   for (int i=1; i <= 11; i++) {
6.       dest[i] = source[i];
7.    }
8.     dest[i] = '\0';
9.       printf("dest = %s", dest);
10. }
```

source character array (ประกาศ ไว้ในบรรทัด 2) คือ 10 ไบต์ของ long แต่ strcpy ( ) (ประกาศไว้บนบรรทัด 3)จะ copy 11 ไบต์ รวมทั้ง null terminator

malloc ( ) ฟังก์ชัน (บรรทัด 4) จะจัดสรรหน่วยความจำ ใน heap ของความยาวของ string ของ source ตามค่า ที่ส่งกลับโดยฟังก์ชัน strlen () ไม่นับรวม null

1) The index value ของ i ใน for (บนบรรทัด 5) เริ่มต้นที่ 1 แต่ ตำแหน่งแรก ในอาร์เรย์ของภาษา C คือ 0
2) The ending condition ของ loop(บนบรรทัด 5) เป็น i < = 11
3) การกำหนด ในบรรทัด ที่ 8 ทำให้เกิดการเขียนออกนอกขอบเขตของการเขียน

2.1.3 Null-Termination Errors

การประกาศค่าคงที่สำหรับ สาม arrray ไม่สามารถจัดสรรได้และการจัดเก็บข้อมูลสำหรับ null-termination เป็นผลให้ ฟังก์ชั่น strcpy (บนบรรทัด 5) เขียน อักขระ null ก่อนจบ array ทั้งนี้ขึ้นอยู่ compiler ที่จะจัดเก็บและจัดสรร storage ซึ่ง null byte อาจ ถูกเขียนทับโดยฟังก์ชัน strcpy ที่ บรรทัด 6 ถ้า เหตุการณ์นี้เกิดขึ้น ในขณะนี้ a ชี้ ไปยัง array ของ 32 ตัวอักษร ในขณะที่ b ชี้ไปยัง array ของ 16 ตัวอักษรและทำการ copy ไปยัง c (บรรทัด 7) ทำให้ฟังก์ชัน strcat ในบรรทัด ที่ 8 ที่จะเขียน ได้เกิน ขอบเขต ของ array

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรม 2.5 Null-termination defect

```
1.int main(int argc, char*argv[]){
2. char a[16];
3. char b[16];
4. char c[32];
5.  strcpy(a, "0123456789abcdef");
6.  strcpy(b, "0123456789abcdef");
7.  strcpy(c, a);
8.  strcat(c, b);
9.  printf("a =%s\n", a);
10. return 0;
```

2.1.4 String Truncation

การตัด string เกิดขึ้นเมื่อ แถวอักขระ ปลายทาง มีขนาดไม่ใหญ่ พอที่จะบรรจุ เนื้อหาของ string การตัด string อาจเกิดขึ้น ในขณะที่อ่านค่าเข้ามาของผู้ใช้ หรือ การคัดลอก string และมักจะเป็นผลมาจาก โปรแกรมเมอร์ พยายามที่จะป้องกัน หน่วยความจำที่เกิดจากการล้นซึ่งเกิดจากการตัด string ซึ่งทำให้เกิดการสูญเสีย ของข้อมูลและ ในบางกรณี สามารถนำไปสู่ ช่องโหว่ของซอฟต์แวร์ได้

String Errors without Functions มีหลาย ฟังก์ชัน ยกตัวอย่างเช่น strcpy(), strcat(), gets(), streadd(), strecpy(), และ strtrns() มันเป็น ไปได้ที่จะ ทำการดำเนินการกับ string ที่ไม่ปลอดภัย ได้โดยไม่ต้อง เรียก ฟังก์ชันซึ่งแสดงให้เห็นถึง ตัวอย่าง โปรแกรม C ที่มี ข้อบกพร่อง ที่เกิดจากการดำเนินการคัดลอก string แต่ไม่ได้มีการเรียกใช้ฟังก์ชันไลบรารี่ของ string ใด ๆ ดังแสดงในรูปต่อไปนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรม 2.6 Defective string manipulation code

```
1.int main(int argc, char*argv[]) {
2. int i =0;
3. char buff[128];
4. char *arg1 =argv[1];
5. while (arg1[i]!="\0"){
6.  buff[i]=arg1[i];
7.  i++;
8.  }
9.  buff[i]='\0';
10. printf("buff =%s\n",buff);
```

2.2 String Vulnerabilities

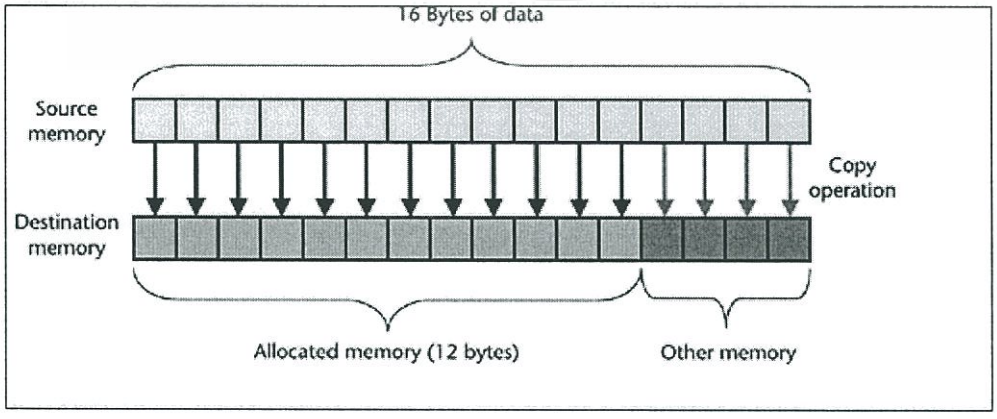
2.2.1 Buffer Overflow

buffer overflow คือการเขียนข้อมูลเกินขอบเขตที่ memory ที่ถูกกำหนดไว้ในโครงสร้างข้อมูลที่มีการจอง ไว้

สาเหตุที่ทำให้เกิด

1) Buffer overflow เกิดจาก เราไม่ได้ ตรวจขอบเขตของ array
2) Standard library functions ไม่ได้มีการตรวจขอบเขตของ array
3) Programmers ไม่ได้ตรวจขอบเขตของ array

*หมายเหตุไม่ใช่ทุก Buffer overflow จะเกิด ช่องโหว่ได้



รูป 2.1 Writing beyond array bounds

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2.2 Stack Smashing

ผู้โจมตีจะผ่าน malicious code และ ก๊อปปี้ของ buffers address ด้วย parameter เพื่อให้เกิดช่องโหว่ผ่าน argument p ฟังชั่น Strcpy จะเกิด overflow เมื่อมีการ return ที่อยู่ของฟังชั่นด้วยที่อยู่ของ buffer ซึ่งบรรจุ shell code การคืนค่าของชุดคำสั่งโดยใช้ที่อยู่ของเป้าหมาย buffer ที่จะ return address และ ไปทำ โค้ดที่ผู้โจมตีต้องการ

โปรแกรม 2.7 ตัวอย่าง Stack Smashing

```
1.void func(char *p,int i){
2.int j=0;
3./* lacal variable */
4.CDummy dummy;
5.char b[128]
6.strcpy(b,p);
```

2.2.3 Code Injection

เป็นการแทรก code เข้าไปในระบบ ซึ่งเมื่อมีผู้ไม่หวังดีแทรกคำสั่งของระบบ (command) เข้าไปเพื่อทำให้ระบบทำงานผิดพลาด หรือการเข้าไปถึงข้อมูลที่ไม่ได้รับอนุญาต

รหัสผ่านที่คุณจะได้รับโปรแกรมที่แสดงในรูป 2.2 จะแสดงการถูกนำไปใช้ประโยชน์ในการเรียกใช้ชุดคำสั่งที่สร้างขึ้นเองได้ เวลานี้โปรแกรมถูกคอมไพล์ของ Red Hat Linux 9.0 โดยใช้ GCC ข้อบกพร่องสามารถเข้าไปในโปรแกรมผ่านไฟล์ข้อมูล Binary (ตามที่แสดงในรูป 2.2) จากไฟล์โดยใช้การเปลี่ยนทิศทางเป็นดังนี้ : % ./BufferOverflow < exploit.bin

```
000  31 32 33 34 35 36 37 38-39 30 31 32 33 34 35 36 "1234567890123456"
010  37 38 39 30 31 32 33 34-35 36 37 38 E0 F9 FF BF "789012345678a. +"
020  31 C0 A3 FF F9 FF BF B0-0B BB 03 FA FF BF B9 FB "1+ú . +□+. +□v"
030  F9 FF BF 8B 15 FF F9 FF-BF CD 80 FF F9 FF BF 31 ". +ï§ . +-ç . +1"
040  31 31 31 2F 75 73 72 2F-62 69 6E 2F 63 61 6C 0A "111/usr/bin/cal "
```

รูป 2.2 Contents of binary file exploit.bin containing shellcode

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ไฟล์ข้อมูลแบบ Binary ที่ไม่สามารถมีบรรทัดใหม่หรืออักขระ NULL จนไบต์สุดท้าย เนื่องมาจากการใช้ประโยชน์จากฟังก์ชันสตริงที่มักจะขึ้นอยู่กับการที่ใช้ฟังก์ชัน gets ( ) โดยการเรียกฟังก์ชัน gets () แปลความหมายเป็นอักขระ Null เป็นสตริงอักขระและการยุติการอ่านข้อมูลบรรทัดใหม่จนกว่าจะมีอักขระหรือเงื่อนไขที่มีการตรวจพบ process

2.2.4 Arc Injection

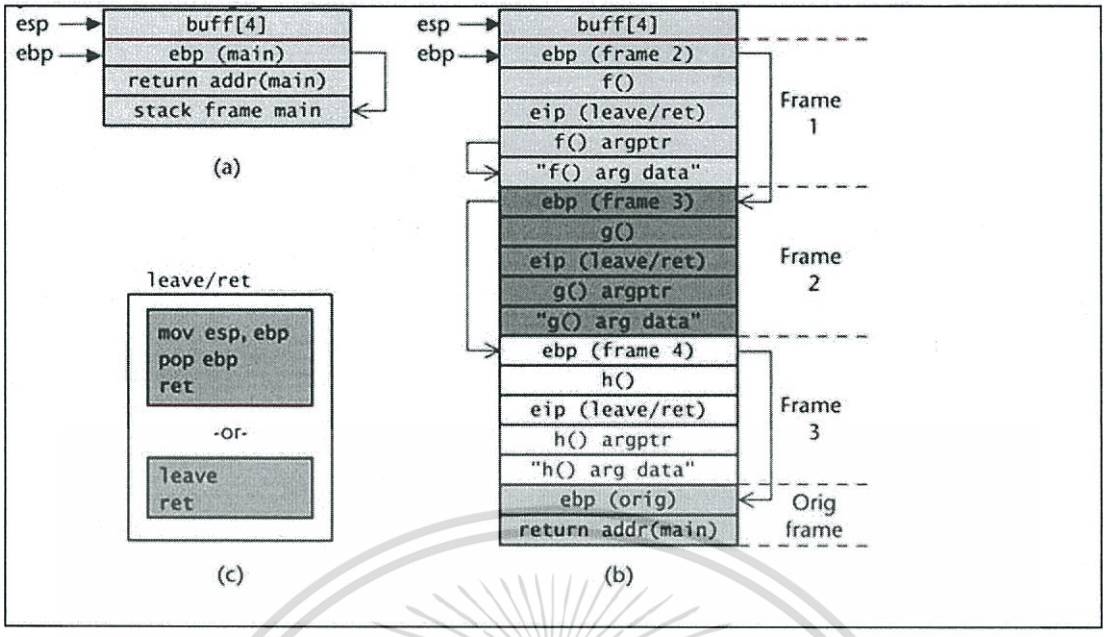
Arc Injection คือ การเปลี่ยนค่า return address ให้กลายเป็นฟังก์ชัน system พร้อมกับระบุค่าอาร์กิวเมนต์ของฟังก์ชันให้เป็นบัฟเฟอร์ที่ใส่เข้าไป ทำให้แฮกเกอร์สามารถเรียกคำสั่งอะไรก็ได้ตามต้องการในเครื่องของเหยื่อการทำ Arc Injection อาจจะเป็นการใช้โค้ดของเหยื่อเอง เช่น ใส่ข้อมูลจนล้นเพื่อไปเปลี่ยนตัวแปรอื่นๆ แต่หากเป็นการเปลี่ยนข้อมูลเพื่อเรียกไลบรารี มักเรียกกันว่าการโจมตีแบบ return-to-libc

โปรแกรม 2.7 มีโปรแกรมที่จะมีความเสี่ยงต่อการเกิดบัฟเฟอร์ล้นได้ ข้อมูลที่จัดมาให้สำหรับผู้ใช้ในการป้อนข้อมูลของผู้ใช้จะถูกคัดลอกไปยังอักขระที่ array ในบรรทัดที่ 4 โดยใช้คำสั่ง memcpy () มีbuffer overflowสามารถเกิดขึ้นได้หากผู้ใช้ ป้อนข้อมูลจะมีขนาดใหญ่กว่าของ buffer รูป 2.3 (a) จะแสดงเนื้อหา stack ก่อนการทำงานฟังก์ชัน get_buff() stack จะประกอบด้วย local variable buff แล้วตามด้วยตัวชี้ และ return กลับมาที่แอดเดรสของ eip สำหรับ main( ) ด้านล่างนี้เป็นจริงสำหรับ stack frame ซึ่งก็คือ main() (อ้างอิงโดยจัดเก็บ frame pointer)

โปรแกรม 2.8 Program vulnerable to arc injection exploit

```
1.#include <string.h>
2.int get_buff(char *user_input){
3. char buff[4];
4. memcpy(buff, user_input,strlen((user_input)+1);
5.  return 0;
6.}
7.int main(int argc,char *argv[]){
8.  get_buff(argv[1]);
9.  return 0;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูป 2.3 Program stack overwritten by binary exploit

2.3(b) จะแสดงเนื้อหาสแตกหลังจากที่ผู้โจมตีจะ overflow เขียนทับเนื้อหาของ stack ได้ ในส่วนนี้ของ stack ยังได้รับการเขียนทับโดย overflow ได้

ผู้โจมตีอาจจะสามารถวางข้อมูล buffer ที่แท้จริงแต่ในตัวอย่างนี้เราจะถือว่า buffer ที่จะถูกเขียนทับด้วยการเติมตัวอักษร ตัวชี้ของเฟรมสำหรับ main() ถูกเขียนทับด้วย frame pointer สำหรับ frame ทั้งหมดนี้จะทำให้ frame ของผู้โจมตีจะเป็นส่วนหนึ่งของการ exploit ได้ เมื่อใช้ exploit ฟังก์ชัน get_buff() ส่งค่ากลับมาให้ซึ่งจะดำเนินการอย่างใดอย่างหนึ่งในสองรูปแบบที่เทียบเท่ากับของ frame ใช้ตัวชี้กลับไปตามลำดับที่แสดงไว้ในรูป 2.3 (c) โดยไม่คำนึงว่าผู้ใช้ใดที่มีการใช้ฟอร์ม , frame pointer (ตอนนี้ที่ชี้ไปยังframeที่ 2) จะถูกย้ายไปในตัวชี้stack การควบคุมการส่งคืนไปยังแอดเดรสบน stack ซึ่งได้รับการเขียนทับด้วยแอดเดรสของฟังก์ชันที่กำหนดเอง ฟังก์ชันนี้จะถูกเรียกใช้และส่งผ่านอาร์กิวเมนต์ที่ติดตั้งไว้บน stack ผู้โจมตีจะต้องระบุหมายเลขที่เหมาะสมและชนิดของอาร์กิวเมนต์จะถูกเรียกใช้โดยถือว่าทำงานได้ ในรูป 2.3 (b) เราจะสันนิษฐานได้ว่าจะยอมรับตัวชี้ไปยังฟังก์ชันเป็น string (ตัวอย่างเช่น " system () ") เนื่องจากเนื้อหาจริงของ string ยังจำเป็นต้องได้รับการจัดหาให้มี string ที่วางอยู่บน stack ของอาร์กิวเมนต์ให้กับฟังก์ชันการใช้งานจริงได้

เมื่อ f ( ) จะส่งกลับให้ พร้อมกับ pop stack eip ที่จัดเก็บและการควบคุมการถ่ายโอนไปยังแอดเดรสนี้แล้ว ในกรณีนี้ได้ถูกเขียนทับด้วย eip แอดเดรสของกลับไปตามลำดับที่แสดงไว้ในรูป 2.3 (c) ซีเควนซ์นี้โดยปกติจะเป็นคำแนะนำที่สร้างขึ้นสำหรับการกลับไปยังฟังก์ชันที่เกิด ช่องโหว่ได้แต่ก็อาจจะปรากฏขึ้นที่ใดก็ได้ในส่วนของรหัสสำหรับการทำงานซึ่งลำดับที่ส่งกลับจะกำหนด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตัวชี้เฟรม (ตอนนี้ที่ชี้ไปยัง frame ที่ 3) ตัวชี้ของ stack และกลับไปที่การควบคุมที่กำหนดเองในครั้งถัดไปที่ฟังก์ชันที่จะถูกเรียก (ในกรณีนี้คือ g ())

ผู้โจมตีจะสามารถทำซ้ำตามลำดับนี้ตามที่จำเป็นในการเรียกใช้ลำดับฟังก์ชันการดำเนินการเพื่อให้บรรลุเป้าหมายที่ใช้ประโยชน์ได้ ผู้โจมตีจะสามารถผลิตซ้ำเนื้อหา frame ต้นฉบับที่ซ้อนกันไปส่งคืนการควบคุมไปยัง main () หลังจากที่มีข้อบกพร่องของการทำงาน

ผู้โจมตีอาจชอบ arc injection เพื่อ injection รหัสผ่านไปได้ด้วยเหตุผลหลายประการเนื่องจาก arc injection ใช้รหัสที่มีอยู่แล้วในหน่วยความจำในระบบเป้าหมายที่ผู้โจมตีจะเป็นเพียงแค่ต้องให้ที่อยู่ของฟังก์ชันและอาร์กิวเมนต์สำหรับการโจมตีที่ประสบความสำเร็จ เขตคลุมสัญญาณสำหรับประเภทนี้ของการโจมตีที่มีขนาดเล็กลงได้อย่างมากและอาจจะใช้เพื่อหาประโยชน์จากความบกพร่องที่ไม่สามารถใช้ในการจู่โจมได้โดยใช้เทคนิคการ injection รหัส arc injection เป็นการโจมตีที่มีการใช้ข้อมูลที่ไม่สามารถแก้ไขได้โดยการทำให้ส่วนของหน่วยความจำ (เช่น stack) nonexecutable ได้

ฟังก์ชันการเรียกเข้าด้วยกันได้อย่างหลากหลายด้วยการทำให้การโจมตีที่มีประสิทธิภาพมากยิ่งขึ้น การรักษาความปลอดภัยเครื่องตั้งโปรแกรมที่คำนึงถึงความคุ้มค่าเป็นหลักตัวอย่างเช่นอาจจะทำตามหลักการให้สิทธิ์น้อยที่สุด และรายการดรอปดาวน์ Saltzer เมื่อไม่จำเป็นต้องใช้สิทธิ์ โดยการใช้ฟังก์ชันต่างๆ ได้อย่างหลากหลายด้วยการเรียกเข้าด้วยกัน , ข้อบกพร่องอาจให้สิทธิพิเศษในการรับสิทธิ์สำหรับตัวอย่างเช่น โดยการเรียกเพื่อติดต่อ setuid () ก่อนติดต่อ กับระบบ

2.3 Mitigation Strategies

ภาษา C ที่ทำสำเนาข้อมูลแต่ไม่ได้มีการตรวจสอบขนาดของข้อมูลเป็นสาเหตุของข้อผิดพลาด (เช่นเดียวกับโปรแกรมเมอร์ที่ใช้ function call เหล่านี้) function call strcat(), strcpy(), sprintf(), vsprintf(), bcopy(), gets() และ scanf() สามารถนำมาใช้ประโยชน์จากช่องโหว่ได้ เนื่องจากฟังก์ชันเหล่านี้ไม่มีการตรวจสอบว่า buffer (พื้นที่ของหน่วยความจำ ใช้สำหรับเก็บข้อมูล) ที่ได้รับการจัดสรรพื้นที่ใน stack มีขนาดใหญ่เพียงพอสำหรับข้อมูลที่จะถูกทำสำเนาลงใน buffer หรือไม่ ขึ้นอยู่กับโปรแกรมเมอร์ว่าจะใช้ฟังก์ชันที่มีการตรวจสอบ(เช่น strncpy())หรือนับจำนวนไบต์ของข้อมูล ก่อนที่จะทำสำเนาลงใน stack หรือไม่ เราจึงมีการใช้ฟังก์ชันเพื่อเข้ามาทดแทน

2.3.1 ฟังก์ชัน strcpy

ความปลอดภัย: ไม่ได้ตรวจสอบการเกิด buffer overflow ขณะที่ทำสำเนาไปยังเป้าหมาย

การแก้ไข: เปลี่ยนไปใช้ฟังก์ชั่น strncpy หรือ strlcpy

2.3.2 ฟังก์ชัน: memcpy

ความปลอดภัย: ไม่ได้ตรวจสอบการเกิด buffer overflow ขณะที่ทำสำเนาไปยังเป้าหมาย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การแก้ไข:ทำให้ตัวแปรที่จะนำมารับค่ามีขนาดพอเพียงสำหรับการรับค่า

2.3.3 ฟังก์ชัน:strcat

ความปลอดภัย:ไม่ได้ไม่ได้ตรวจสอบการเกิด buffer overflow ขณะที่ทำสำเนาไปยังเป้าหมาย

การแก้ไข:เปลี่ยนไปใช้ฟังชั่น strcat หรือ strlcat แทน

2.3.4 ฟังก์ชัน strncpy

ความปลอดภัย: ง่ายต่อการใช้ไม่ถูกต้องไม่ได้ทำการปิดด้วย \0 เสมอไป หรือตรวจสอบตัวชี้ที่ไม่ถูกต้อง

2.3.5 ฟังก์ชัน: strncat

ความปลอดภัย: ง่ายต่อการใช้งานที่ไม่ถูกต้อง(เช่น การคำนวณค่าสูงสุดที่ตัวแปรสามารถรับได้)

การแก้ไข:ใช้ฟังก์ชั่น strlcat หรือคำนวณค่าสูงสุดที่ตัวแปรรับได้ให้ถูกต้อง

2.3.6 ฟังก์ชัน:get,_getts

ความปลอดภัย: ไม่ได้ตรวจสอบการเกิด buffer overflow

การแก้ไข:ใช้ฟังก์ชัน fgets แทน

2.4 Pointer Subterfuge Vulnerabilities

นอกจากกระบวนการ "เติม" ข้อมูลเพื่อให้โปรแกรมทำงานตามที่แฮกเกอร์ต้องการแล้ว ยังมีอีกกระบวนการหนึ่ง คือการเปลี่ยนการทำงานของโปรแกรมด้วยการเปลี่ยนพอยเตอร์ ที่เรียกกระบวนการนี้ว่า Pointer Subterfuge ทำได้ในกรณีที่หน่วยความจำที่ต่อท้ายบัฟเฟอร์เป็นพอยเตอร์ไปยังฟังก์ชั่น กรณีเช่นนี้แฮกเกอร์สามารถเข้าไปกำหนดจุดที่ฟังก์ชันจะถูกเรียกได้ ฟังก์ชั่นตัวอย่างแสดงให้เห็นฟังก์ชันที่เสียงต่อการถูกเปลี่ยนค่าในพอยเตอร์ไปยังฟังก์ชั่น หากแฮกเกอร์สามารถรู้ได้ว่าฟังก์ชั่นที่ต้องการนั้นอยู่ที่พื้นที่หน่วยความจำใด ก็สามารถเรียกฟังก์ชั่นนั้นได้ตามใจชอบ กระบวนการ Pointer Subterfuge นั้นแบ่งออกเป็นแบบย่อยๆ ได้อีกสามแบบ ได้แก่ Function Pointer Subterfuge ที่แสดงในตัวอย่าง, Data Pointer Subterfuge ที่เป็นการเปลี่ยนจุดชี้ข้อมูล ทำให้โปรแกรมทำงานตามที่ต้องการ คล้ายกับการทำ Arc Injection, และ VPTR Smashing ที่เป็นรูปแบบเฉพาะของภาษา C++ ที่ทุกอ็อบเจกต์จะมีพอยเตอร์ไปยังตาราง virtual function (VTBL) เพราะแต่ละอ็อบเจกต์นั้นอาจจะมีฟังก์ชั่นที่ต่างกันไป หากแฮกเกอร์สามารถย้ายจุดชี้ไปยังตารางนี้ได้ ก็เท่ากับสามารถทำให้โปรแกรมเรียกฟังก์ชันใดก็ได้ตามต้องการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปฏิบัติการ ยูนิกซ์ มีทั้ง ข้อมูล และ ส่วน BSS ประกอบด้วย Global Variableและ Static Variableที่มีค่าคงที่เป็นศูนย์โดยอัตโนมัติจะถูกเก็บไว้ในส่วนนี้ โดยสามารถเรียกส่วนนี้ได้อีกกว่า ส่วนของตัวแปรที่มีค่าเป็นศูนย์โดยแต่ละ Process จะมีส่วนแยกกัน

โปรแกรม 2.9 Data declarations and process memory organization

```
1.static int GLOBAL_INIT =1; /*data segment, global */
2.static int global_uninit; /*BSS segment, global */
3.
4.int main(int argc, char *argv[]){ /*stack, local */
5. int local_init =1; /*stack, local */
6. int local_uninit; /*stack, local */
7. static int local_static_init =1; /*data seg, local */
8. static int local_static_uninit; /*BSS segment, local */
  /*storage for buff_ptr is stack, local */
  /*allocated memory is heap, local */
9. int *buff_ptr =(int *)malloc(32);
10.}
```

2.4.1Function Pointers

function pointer เป็น pointer ที่ใช้สำหรับการ dynamic bind ตัวแปร f ให้เป็น function ใดๆ ก็ได้ เช่นการ bind f ให้เป็น printf

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรม 2.10 Data declarations and process memory organization

```
1.void good_function(const char *str){…}
2.int main(int argc, char *argv[]){
3. static char buff[BUFFSIZE];
4. static void (*funcPtr)(const char *str);
5. funcPtr =&good_function;
6. strncpy(buff, argv[1], strlen(argv[1]));
7. (void)(*funcPtr)(argv[2]);
```

2.4.2 Virtual Pointers

C++ ช่วยให้คำจำกัดความของการทำงานเสมือนจริง ฟังก์ชันเสมือนเป็นฟังก์ชันที่สมาชิกของคลาสที่ได้รับการประกาศว่าการใช้คำสำคัญในระบบเสมือน ฟังก์ชันการทำงานอาจถูกแทนที่ด้วยฟังก์ชันที่มีชื่อเหมือนกันในคลาสที่ได้รับมา ตัวชี้ไปยังวัตถุคลาสที่ได้รับมาอาจถูกมอบหมายให้กับที่พักในระดับ first class และตัวชี้ให้ฟังก์ชันที่เรียกว่าผ่านตัวชี้ ไม่มีฟังก์ชันการทำงานเสมือนจริงที่มีการเรียกฟังก์ชันเรียกคลาสได้เนื่องจากมีการเชื่อมโยงกับ static pointer เมื่อใช้ฟังก์ชันการทำงานเสมือนจริงให้คลาสที่นำมาใช้เป็นฟังก์ชันที่เรียกว่ามีการเชื่อมโยงกับประเภทแบบไดนามิกของอ็อบเจกต์

โปรแกรม 2.11 แสดงให้เห็นถึงการทำงาน Semantic ของระบบเสมือนจริงที่ คลาสจะถูกกำหนดให้เป็นฐานของคลาสและประกอบด้วยฟังก์ชัน f() และ ฟังก์ชันเสมือน g ( ) Class B คือได้มาจากทั้ง a และ override สองฟังก์ชัน

ตัวชี้ my_b มีฐานที่มีการประกาศคลาสใน Main () แต่ถูกกำหนดค่าโดยอ็อบเจกต์ของคลาส B ที่นำมาใช้ได้ เมื่อใช้ฟังก์ชัน nonvirtual my_b->f() จะถูกเรียกในบรรทัดที่ 21 ให้ใช้ฟังก์ชัน f ( ) ที่เกี่ยวข้องกับ (คลาสหลัก) จะถูกเรียกได้ เมื่อใช้ฟังก์ชันเสมือน my_b->f() จะถูกเรียกในบรรทัดที่ 22 ฟังก์ชัน g() ที่เกี่ยวข้องกับการที่ได้รับมาคลาส B จะถูกเรียกว่า the derived class

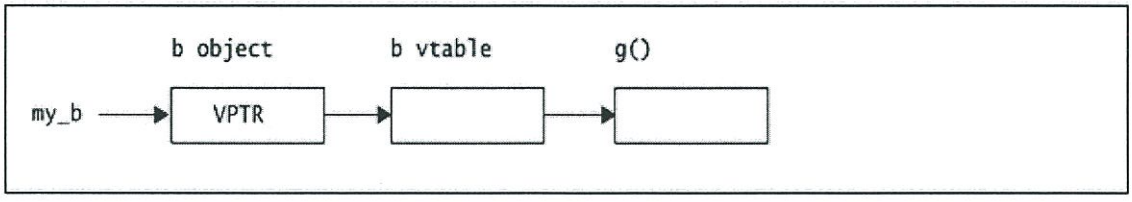
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรม 2.11 The semantics of virtual functions

```
1.class a {
2. public:
3.  void f(void){
4.   cout << "base f"<< endl;
5.  };
6.  virtual void g(void){
7.   cout << "base g"<< endl;
8.  };
9.};
10.class b:public a {
11. public:
12.  void f(void){
13.   cout << "derived f"<< endl;
14.  };
15.  void g(void){
16.   cout << "derived g"<< endl;
17.    };
18.};
19.int _tmain(int argc, _TCHAR*argv[]){
20. a *my_b =new b();
21. my_b->f();
22. my_b->g();
```

C++ คอมไพเลอร์ส่วนใหญ่นำไปใช้โดยใช้ฟังก์ชันเสมือนฟังก์ชันเสมือนตาราง VTBL ที่เป็นอาร์เรย์ของ พอยน์เตอร์ VTBL ฟังก์ชันที่ถูกใช้งานที่จัดส่ง Runtime สำหรับฟังก์ชันเสมือนการเรียก อ็อบเจกต์แต่ละอันใน VTBL ผ่านตัวชี้เสมือน VPTR ในส่วนหัวของอ็อบเจกต์ VTBL จะมีตัวชี้ไปยังแต่ละการนำไปใช้งานของฟังก์ชันเสมือนได้ รูป 2.4 แสดงโครงสร้างข้อมูลที่ได้จากตัวอย่าง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูป 2.4 VTBL runtime representation

มีความเป็นไปได้ที่จะเขียนทับในฟังก์ชันพอยน์เตอร์ VTBL หรือเพื่อเปลี่ยนไปยังจุดเชื่อมต่ออื่นเพื่อ VPTR และ VTBL ที่กำหนดเองได้ ซึ่งสามารถทำได้โดยการเขียนหรือทำให้เกิดการล้นของข้อมูลโดยตรงลงในออบเจกต์ได้ ใน บัฟเฟอร์ที่จะเขียนทับ VTBL และ VPTR ของวัตถุ และทำให้ผู้โจมตีจะทำให้เกิดการเรียกใช้ชุดคำสั่งฟังก์ชันพอยน์เตอร์ VPTR

Attacker สามารถ overwrite function pointers ใน VTBL ได้หรือ เปลี่ยน VPTR ชี้ไปยัง VTBL ใหม่ Attacker ยังสามารถใช้ arbitrary memory write หรือ buffer overflow โดยตรงบน the object.

2.5 Mitigation Strategies

2.5.1 W^X

ลดกระบวนการ ที่มีช่องโหว่ส่วนหน่วยความจำ สามารถเป็นได้ทั้ง ที่สามารถเขียนได้ หรือปฏิบัติการ แต่ไม่ใช่ทั้งสองไม่ได้ทำงานกับ เป้าหมาย เช่น atexit ( ) ที่ จะต้องมีการ เขียนได้ ที่รันไทม์ และ ปฏิบัติการ ทั้งสอง

Canaries

ป้องกันการ ล้น บัฟเฟอร์ บน stack และ เขียนทับ ตัวชี้ stack ที่มีการป้องกัน ไม่ป้องกันการ Overflowing ของ stack ด้วยตัวมันเอง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

2.6 Integer Vulnerabilities

2.6.1 Integer Conversions

การแปรชนิดของตัวเลขใน C และ C ++ นั้น จะยึดหลักการตาม C99 Standard rules โดย C99 Standard rules กำหนดค่าสูงสุดและค่าต่ำสุดเอาไว้ ตามรูปต่อไปนี้

Constant	C99 minimum value	Visual C++ .NET and GNU CC on Intel x86	Description
CHAR_BIT	8	8	Number of bits for smallest object that is not a bit-field (byte)
SCHAR_MIN	-127 // $-(2^7 - 1)$	-128	Minimum value for an object of type signed char
SCHAR_MAX	+127 // $2^7 - 1$	+127	Maximum value for an object of type signed char
UCHAR_MAX	255 // $2^8 - 1$	255	Maximum value for an object of type unsigned char
SHRT_MIN	-32,767 // $-(2^{15} - 1)$	-32,768	Minimum value for an object of type short int
SHRT_MAX	+32,767 // $2^{15} - 1$	+32,767	Maximum value for an object of type short int
USHRT_MAX	65,535 // $2^{16} - 1$	65,535	Maximum value for an object of type unsigned short int
INT_MIN	-32,767 // $-(2^{15} - 1)$	-2,147,483,648	Minimum value for an object of type int
INT_MAX	+32,767 // $2^{15} - 1$	+2,147,483,647	Maximum value for an object of type int
UINT_MAX	65,535 // $2^{16} - 1$	4,294,967,295	Maximum value for an object of type unsigned int
LONG_MIN	-2147483647 // $-(2^{31} - 1)$	-2,147,483,648	Minimum value for an object of type long int
LONG_MAX	+2,147,483,647 // $2^{31} - 1$	+2,147,483,647	Maximum value for an object of type long int
ULONG_MAX	4,294,967,295 // $2^{32} - 1$	4,294,967,295	Maximum value for an object of type unsigned long int
LLONG_MIN[a]	-9223372036854775807 // $-(2^{63} - 1)$	-9223372036854775808	Minimum value for an object of type long long int
LLONG_MAX[b]	+9223372036854775807 // $2^{63} - 1$	+9223372036854775807	Maximum value for an object of type long long int
ULLONG_MAX[c]	18446744073709551615 // $2^{64} - 1$	18446744073709551615	Maximum value for an object of type unsigned long long int

รูป 2.5 Maximun and Minimum Extents of Integer Types

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้
144431

2.6.2 Interger Promotions

จากโปรแกรม 2.12 Interger Promotion จะช่วยในการหลีกเลี่ยงข้อพลาดที่อาจเกิดจากการคำนวณล้นของข้อมูลได้ ซึ่งบรรทัด 5 ถึง บรรทัดที่ 7 แสดง ค่า c1 ที่บวกกับค่า c2 และ ผลของการบวกนี้จะถูกรวมไปบวกกับค่า c3 ซึ่งการบวก c1 และ c2 จะเป็นผลที่ทำให้เกิด overflow ของ signed char เพราะผลของการบวกเป็นค่าสูงสุดของ signed char แล้ว อย่างไรก็ตาม c1,c2,c3 แต่ละตัวแปรจะถูกแปลงค่าไปเป็น Interger และผลลัพธ์ของค่านั้นถูกตัดทอนและถูกเก็บไว้ในตัวแปร cresult เพราะว่าผลลัพธ์นั้นอยู่ในช่วงของชนิด signed char และ การตัดนั้นจะไม่ทำให้ผลลัพธ์ของข้อมูลนั้นสูญหายไป

โปรแกรม 2.12 Preventing arithmetic errors with implicit conversions

```
1.char cresult, c1, c2, c3;
2.c1 =100;
3.c2 =90;
4.c3 =-120;
5.cresult =c1 +c2 +c3;
```

2.6.3 Integer Conversion Rank

ชนิดของ integer ทุกชนิดนั้นมี Integer Conversion Rank ที่กำหนดการดำเนินการต่างๆ เอาไว้ กฎต่อไปนี้สำหรับใช้ในการกำหนดตำแหน่งในการแปลงจำนวนเต็มที่ได้กำหนดไว้ใน C99 Standard rules มีดังต่อไปนี้

1) ต้องไม่มี signed integer ที่แตกต่างกัน 2 ชนิด ถึงแม้ว่าจะมีการแทนค่าที่เหมือนกัน
2) การจัดลำดับของชนิดของ signed integer จะสูงกว่าตำแหน่งของชนิด signed integer ใดๆ ที่มีความแม่นยำน้อยกว่า
3) การจัดลำดับ long long int จะสูงกว่า การจัดลำดับ long int และจะสูงกว่า int , short int , signed char ด้วย
4) การจัดลำดับใดๆของ unsigned integer จะเทียบเท่ากับ การจัดอับดับของ signed integer ที่ตรงกัน(ถ้ามี)
5) การจัดอับดับใดๆของ standard integer จะสูงกว่าการจัดอันดับ extend integer ใดๆที่มีความกว้างเท่ากัน
6) การจัดอันดับใน char เทียบเท่าการจัดอับดับ signed char และ unsigned char7) การจัดอันดับใดๆของ extend signed integer เมื่อเทียบกับ extended signed integer

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ชนิดอื่นที่มีความแม่นยำเดียวกันกับที่มีการ implementation ไว้แล้วจะยังคงอยู่ภายใต้กฎของ integer conversion rank อยู่

7) สำหรับ ชนิดของ integer ทั้งหมดของ T1,T2 และ T3 , ถ้า T1 มีการจัดอันดับสูงกว่า T2 และ T2 มีจัดอันดับสูงกว่า T3 แล้ว T1 จะมีการจัดอันดับสูงกว่า T3 ด้วย

Integer conversion rank จะใช้ usual arithmetic conversions ที่กำหนดไว้ซึ่งต้องการรองรับการทำ mixed integer ด้วย

2.6.4 Conversions from Unsigned Integer Types

การแปลงที่เกิดขึ้นระหว่าง signed และ unsigned integer ขนาดใดๆ และอาจส่งผลทำให้เกิดการสูญหายของข้อมูลหรือการปัญหาการตีความของข้อมูลเมื่อค่านั้นไม่สามารถแทนที่ด้วยค่าชนิดใหม่ได้นั้นเองการแปลงค่าของชนิด unsigned integer ที่มีขนาดเล็ก ไปเป็น ค่าของชนิด unsigned integer ที่มีขนาดใหญ่จะต้องมีการระมัดระวังและมักจะเกิดการขยายขนาดของค่านั้น

เมื่อ unsigned integer ที่มีขนาดใหญ่ ถูกแปลงไปเป็นค่าของ unsigned integer ที่มีขนาดเล็ก ซึ่งค่าที่มีขนาดใหญ่จะถูกตัดทอนและจะมีการเก็บรักษา low-order เอาไว้ เมื่อค่าของ unsigned integer ที่มีขนาดใหญ่ถูกแปลงไปเป็น signed integer ที่มีขนาดเล็ก ซึ่งค่านี้จะถูกตัดทอนและ high-order bit จะถูกเก็บค่าของ sign bit ไว้ ซึ่งทั้งสองกรณีนี้ข้อมูลก็อาจสูญหายไปถ้าค่านั้นไม่สามารถแทนในค่าใหม่ได้

เมื่อ unsigned integer ถูกแปลงไปเป็น ชนิดของ singed integer (ตัวอย่าง การแปลง unsigned char ไปเป็น char) bit pattern จะถูกเก็บรักษาเอาไว้ จากรูปจะแสดงการแปลงค่าชนิด unsigned integer โดยสีเทาแสดงถึงผลลัพธ์ที่เกิดจากการแปลงค่าทำให้เกิดข้อมูลสูญหายได้และสีเทาดำจะแสดงการแทนค่าที่ไม่ถูกต้อง

โปรแกรม 2.13 Integer promotion error

```
1.unsigned int l =ULONG_MAX;
2.char c =-1;
3.if (c ==l){
4. printf("Why is -1 =4,294,967,295???\n");
5.}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

From	To	Method
unsigned char	char	Preserve bit pattern; high-order bit becomes sign bit
unsigned char	short	Zero-extend
unsigned char	long	Zero-extend
unsigned char	unsigned short	Zero-extend
unsigned char	unsigned long	Zero-extend
unsigned short	char	Preserve low-order byte
unsigned short	short	Preserve bit pattern; high-order bit becomes sign bit
unsigned short	long	Zero-extend
unsigned short	unsigned char	Preserve low-order byte
unsigned long	char	Preserve low-order byte
unsigned long	short	Preserve low-order word
unsigned long	long	Preserve bit pattern; high-order bit becomes sign bit
unsigned long	unsigned char	Preserve low-order byte
unsigned long	unsigned short	Preserve low-order word

รูป 2.6 Conversions from Unsigned Integer Type

2.6.5 Conversions from Signed Integer Types

เมื่อ signed integer ถูกแปลงไปเป็น unsigned integer ที่มีขนาดเท่ากันหรือมากกว่าและค่าของ signed integer ไม่เป็นค่าลบและค่านั้นไม่ได้ถูกเปลี่ยนแปลงและ การแปลง shorter signed integer จะเป็นการตัดทอน high –order bit เมื่อชนิดของ signed integer ถูกแปลงไปเป็น unsigned ข้อมูลนี้จะไม่สูญหาย เพราะ จะใช้ bit pattern ในการรักษาข้อมูลไว้ อย่างไรก็ตาม high-order bit จะสูญหายไปใน ฟังก์ชัน sign bit ถ้าค่าของ signed integer ไม่เป็นค่าลบซึ่งค่านั้นจะต้องไม่มีการเปลี่ยนแปลง แต่ ถ้าค่านั้นเป็นลบ ผลลัพธ์ของ unsigned จะมีขนาดใหญ่ ดังตัวอย่างในรูป 2.6 บรรทัด 3 จะเป็นการเทียบค่า ของ c กับ 1 เนื่องจาก integer promotions c จะถูกแปลงไปเป็น unsigned integer ซึ่งมีค่าอยู่ระหว่าง 0xFFFFFFFF หรือ 4,294,967,295

จากรูป 2.7 จะเป็นตารางสรุปการแปลงค่าของชนิด signed integer ซึ่งสีเทาจะแสดงการสูญหายของข้อมูลและสีเทาดำจะแทนการแทนค่าที่ไม่ถูกต้อง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

From	To	Method
char	short	Sign-extend
char	long	Sign-extend
char	unsigned char	Preserve pattern; high-order bit loses function as sign bit
char	unsigned short	Sign-extend to `short`; convert `short` to `unsigned short`
char	unsigned long	Sign-extend to `long`; convert `long` to `unsigned long`
short	char	Preserve low-order byte
short	long	Sign-extend
short	unsigned char	Preserve low-order byte
short	unsigned short	Preserve bit pattern; high-order bit loses function as sign bit
short	unsigned long	Sign-extend to `long`; convert long to `unsigned long`
long	char	Preserve low-order byte
long	short	Preserve low-order word
long	unsigned char	Preserve low-order byte
long	unsigned short	Preserve low-order word
long	unsigned long	Preserve bit pattern; high-order bit loses function as sign bit

รูป 2.7 Conversions from Signed Integer Type

2.6.6 Integer Error Conditions

Integer overflow มักจะเกิดขึ้นเมื่อ จำนวน integer มีค่าเกินกว่าค่าสูงสุดของค่า integer ที่กำหนดไว้หรือลดลงเกินกว่าค่าต่ำสุด ซึ่ง integer overflow จะเกี่ยวข้องกับการแทนค่าข้อมูลพื้นฐานด้วย Overflow นั้นสามารถเป็นได้ทั้ง signed หรือ unsigned ซึ่ง signed overflow จะเกิดขึ้นเมื่อมีการดำเนินการไปยัง signed bit ส่วน unsigned overflow จะเกิดขึ้นเมื่อไม่สามารถแทนค่าพื้นฐานนั้นได้ จาก โปรแกรม 2.14 แสดงให้เห็นถึงผลกระทบของเกิด overflow ใน signed integer และ unsigned integer ซึ่ง signed integer i ถูกใส่ค่าไว้สูงสุดที่ 2,147,483,647 ในบรรทัดที่ 3 และเพิ่มขึ้นค่า ในบรรทัด 4. ผลการดำเนินงานจะทำให้เกิด integer overflow เกิดขึ้น และ เมื่อ i ถูกกำหนดค่าเป็น -2,147,483,648 (ซึ่งเป็นค่าต่ำสุดสำหรับชนิดข้อมูล int) ผลของการดำเนินการก็จะเป็น (2,147,483,647 + 1 = -2,147,483,648) จะเห็นได้ชัดว่ามีข้อผิดพลาดทางคณิตศาสตร์เกิดขึ้น โดย Integer overflow นอกจากจะเกิดขึ้นเมื่อมีการเพิ่มของ unsigned integer ที่มีค่าสูงสุด (บรรทัด 6-8) และ decrementing signed integer ที่มีค่าต่ำสุดแล้ว (บรรทัด 9-11) หรือ decrementing unsigned integer ที่ค่าต่ำสุดที่แสดงไว้บนบรรทัด 12 -14

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรม 2.14 Signed and unsigned integer overflows

```
1.int i;
2.unsigned int j;
3.i =INT_MAX;  //2,147,483,647
4.i++;
5.printf("i =%d\n", i); /*i =-2,147,483,648 */
6.j =UINT_MAX; //4,294,967,295;
7.j++;
8.printf("j =%u\n", j); /*j =0 */
9.i =INT_MIN; //-2,147,483,648;
10.i--;
11.printf("i =%d\n", i); /*i =2,147,483,647 */
12.j =0;
13.j--;
```

Sign Error เกิดขึ้นเมื่อมีการแปลงจาก signed integer ไปเป็น unsigned integer เมื่อ signed integer ถูกแปลงไปเป็น unsigned integer ที่มีขนาดเท่ากัน bit pattern จะถูกเก็บรักษาไว้ และเมื่อ signed integer ถูกแปลงไปเป็น unsigned integer ด้วยค่าที่มีขนาดใหญ่กว่า จะมีการขยายค่านั้นก่อนที่จะมีการแปลงค่า ซึ่งทั้งสองกรณีนี่ high-order bit จะถูกตัดออกไปเป็น signed bit ถ้าค่าของ signed integer ไม่เป็นค่าลบ ค่านั้นก็จะไม่เปลี่ยนแปลง อย่างไรก็ตามเมื่อค่าของ signed integer เป็นค่าลบผลที่ได้มักจะเป็นค่าบวกที่มีขนาดใหญ่กว่าค่าเดิม ซึ่งแสดงตามโปรแกรม 2.15

โปรแกรม 2.15 Sign error

```
1. int i =-3;
2. unsigned short u;
3. u =i;
4. printf("u =%hu\n", u);  /*u =65533 */
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Truncation Errors เกิดขึ้นเมื่อ integer ถูกแปลงเป็น integer ที่มีขนาดเล็กและค่าดั้งเดิมของ integer จะอยู่นอกช่วงของชนิดข้อมูล integer ที่มีขนาดเล็กนั้นและ ปกติ low-order bit ค่าเดิมจะถูกเก็บรักษาไว้และhigh-order bit ก็จะหายไป เมื่อค่า unsinged ถูกแปลงไปเป็นค่า signed ที่มีความยาวเท่ากัน ตัว bit pattern จะถูกเก็บรักษาไว้ซึ่งเป็นสาเหตุที่ high-order bit จะถูกแปลงไปเป็น sign bit จากผลดังกล่าวทำให้ ค่าสูงสุดของ signed integer จะถูกแปลงไปเป็นค่าลบ ดังแสดงในโปรแกรม 2.16 แสดง Truncation Errors และไม่เกิด signed error เพราะข้อมูลมีชนิด signed ซึ่งมีหนึ่งบิตน้อยกว่าที่จะแทนในผลลัพธ์ได้

โปรแกรม 2.16 Integer truncation error

```
1. unsigned short int u =32768;
2. short int i;
3. i =u;
4. printf("i =%d\n", i); /* i =-32768 */
5. u =65535;
6. i =u;
7. printf("i =%d\n", i); /* i =-1 */
```

2.6.7 Integer Operations

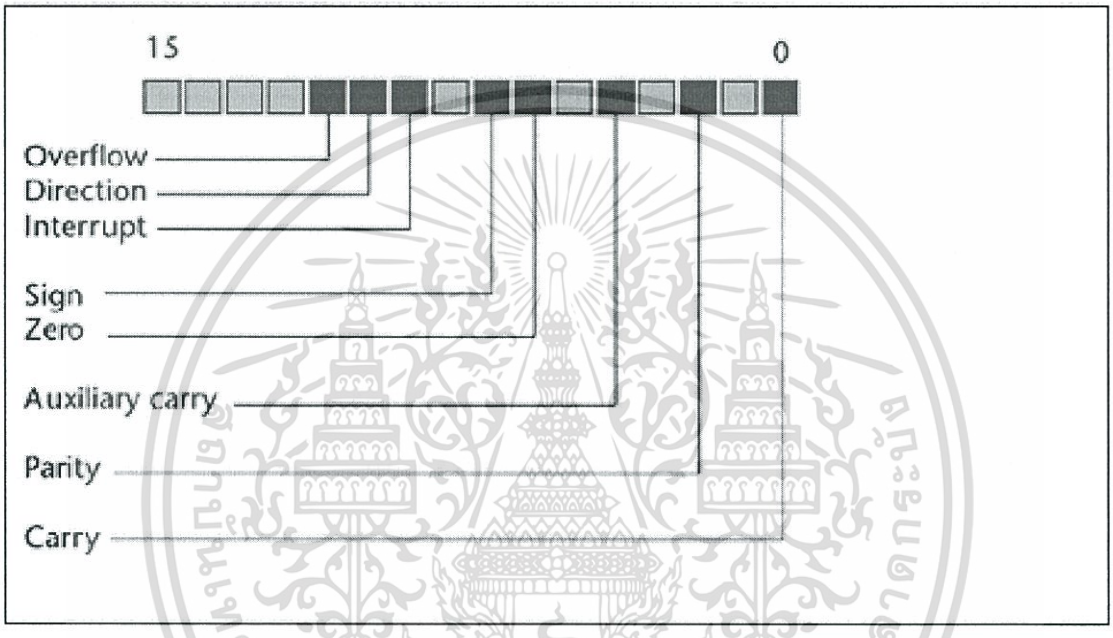
Integer Operations สามารถส่งผลให้เกิดเงื่อนไขพิเศษ ซึ่งผลลัพธ์ของการ operation กันนั้นอาจจะทำให้เกิดค่าที่ไม่คาดคิดได้ ซึ่งค่าที่ไม่คาดคิดนั้นอาจจะทำให้เกิดพฤติกรรมที่ไม่คาดคิดในการเขียนโปรแกรมและช่องโหว่เรื่องความปลอดภัยได้ ซึ่งสามารถแบ่งการ operation ออกเป็นดังนี้

การบวก Integer อาจจะเป็นผลให้เกิด overflow ได้ซึ่งส่งผลให้ไม่สามารถแทนจำนวนบิตที่จองไว้ได้ใน integer ได้ จำนวนที่เกิดขึ้นจากจำนวนบิตที่ถูกจองไว้จะขึ้นอยู่กับสถาปัตยกรรมของเครื่องและชนิดของ operation นั่นๆ ตัวอย่างเช่น สมมติเอา char 2 ตัวมาบวกกันซึ่งสถาปัตยกรรมของเครื่องแทน char ด้วย 8 bit และ 16 bit และ int ใช้ 32 bit และ long long ใช้ 64 bit เนื่องจาก integer promotions ซึ่ง operation ของ char 2 ตัว จะถูกแปลงไปเป็น signed int จากเดิมที่เป็น signed char ก่อนที่จะนำมาบวกกัน แต่จะไม่เกิด overflow ขึ้น เพราะ SCHAR_MAX + SCHAR_MAX จะน้อยกว่า INT_MAX และ SCHAR_MIN + SCHAR_MIN จะมากกว่า INT_MIN อย่างไรก็ตาม ยังไม่มีการรับประกันว่า จะมีค่าที่ได้มาจากบวกกันเป็นประเภท int หรือ long long เพราะ integer promotions ไม่ได้มีการดำเนินการไว้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เป็นไปได้ที่ SCHAR_MAX + SCHAR_MAX จะเกินขนาดของ signed char เป็นผลให้เกิด Truncation Errors ระหว่างการทำการบวกกันได้ อย่างไรก็ตาม จะไม่เกิด overflow ระหว่างการบวกกันและ

Error Detection Signed และ unsigned overflow เป็นผลมาจากการทำการบวกกันที่ถูกตรวจพบและรายงานใน IA-32 รูป 2.8 แสดงชุดคำสั่ง IA-32 และ ชุดคำสั่งการ บวก และแสดง flag ที่เกี่ยวข้องกับ overflow



รูป 2.8 Layout of the flags register [Intel 04]

1) Overflow flag จะระบุ Signed Arithmetic Overflow
2) Carry flag จะระบุ Unsigned Arithmetic Overflow

โปรแกรม 2.17 แสดงชุดคำสั่งของภาษา assembly ที่สร้างจากคอมไพเลอร์ของ Visual C++ .NET บรรทัดที่1-3 เสดงให้เห็นการบวกกันของ sign characters บรรทัดที่ 4-6 การบวกกันของ unsign characters ทั้งสอง operands ในแต่ละกรณีจะมีการใส่ค่าลงใน register ชนิด 32 bit (eax or ecx) ซึ่งส่งผลให้ค่ามีขนาด 32 บิต (ซึ่งเป็นขนาดของ int) บรรทัดที่ 7-8 แสดงให้เห็นการบวก unsigned int ซึ่งค่าจะมีขนาดเท่ากับ 32 บิต(ดังนั้นจะเป็น pointer dowrd) บรรทัดที่ 9-10 แสดงให้ถึงการบวกของ long long ซึ่งคำสั่งการ บวก จะเพิ่ม low-order 32 bit และ คำสั่ง abc จะเพิ่ม high-order 32 bit และ ค่าของ carry bit เข้าไปด้วยนั่นเอง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรม 2.17 I Assembly code generated by Visual C++ for addition

```
sc1 +sc2
 1.movsx         eax, byte ptr [sc1]
 2.movsx         ecx, byte ptr [sc2]
 3.add           eax, ecx
uc1 +uc2
 4.movzx         eax, byte ptr [uc1]
 5.movzx         ecx, byte ptr [uc2]
 6.add           eax, ecx
ui1 +ui2
 7.mov           eax, dword ptr [ui1]
 8.add           eax, dword ptr [ui2]
sll1 +sll2
 9.mov          eax, dword ptr [sll1]
10.add          eax, dword ptr [sll2]
```

ก่อนเงื่อนไขของการบวกของ integer สามารถทำให้เกิด overflow ได้ ถ้า ผลรวมของ left-head side (LHS) และ right-hand side (RHS) ของการบวกมีค่ามากกว่า UINT_MAX ของการบวก int และ ULLONG_MAX ของการบวก unsigned long long

รูป 2.9 แสดงการบวกของ signed integer และเงื่อนไขการเกิด overflow แบบต่างๆ

LHS	RHS	Exceptional condition
Positive	Positive	Overflow if `INT_MAX - LHS <= RHS`
Positive	Negative	None possible
Negative	Positive	None possible
Negative	Negative	Overflow if `LHS <= INT_MIN - RHS`

รูป 2.9 Addition of Sign Integers of Type int

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Error Detection ชุดคำสั่งของ IA-32 ที่ใช้ในการลบ คือ sub และ sbb ใช้เป็นตัวยืมในการลบ ชุดคำสั่งในการลบ ชุดของคำสั่งลบของตัวถูกดำเนินการที่ 2 (ที่มาจากตัวดำเนินการ) จากตัวถูกดำเนินการครั้งแรก (ตัวที่ถูกดำเนินการปลายทาง) และ จะเก็บผลลัพธ์ไว้ในตัวดำเนินการปลายทาง ซึ่ง ตัวดำเนินการปลายทางนั้นสามารถเก็บใน register หรือ เก็บไว้บน memory

โปรแกรม 2.18 จะแสดงการใช้คำสั่ง sub และ sbb ในการลบจำนวนเต็ม s112 ของ signed long long กับ จำนวน เต็ม s111 ของ signed long long

โปรแกรม 2.18 Assembly code generated by Visual C++ for subtraction

```
s111 -s112
1.mov eax, dword ptr [s111]
2.sub eax, dword ptr [s112]
3.mov ecx, dword ptr [ebp-0E0h]
4.sbb ecx, dword ptr [ebp-0F0h]
```

ก่อนเงื่อนไขในการทดสอบ overflow ของ unsigned integer จะทำการพิจารณาจาก LHS <RHS

สำหรับ singed integer ของตัวเลขผสม จะมีเงื่อนไขดังนี้

ถ้า LHS ที่เป็นจำนวนลบ และ RHS เป็นจำนวน บวก จะเช็คโดยใช้ lhs >= INT_MAX+rhs

ถ้า LHS ที่ไม่เป็นจำนวนลบ และ RHS ที่เป็นจำนวนลบ จะเช็คโดยใช้ lhs <= INT_MAX+rhs

ยกตัวอย่าง เอา 0 - INT_MIN จะทำให้เกิดเงื่อนไข overflow เพราะ ผลรับของ operation นั้นมากกว่า ค่าสูงสุดที่จะใช้แทนได้

หลังเงื่อนไขการทดสอบ overflow ของ signed integer จะเอา difference = lhs –rhs ดังตัวอย่างต่อไปนี้

ถ้า rhs มีค่าไม่เป็นลบ และ difference > lhs จะเกิด overflow เกิดขึ้นได้

ถ้า rhs มีค่าลบ และ difference < lhs จะเกิด overflow เกิดขึ้นได้

ใน case อื่นๆ จะไม่เกิด overflow เกิดขึ้นและสำหรับ unsigned integer จะเกิด overflow ถ้า difference มีค่ามากกว่า lhs

การคูณอาจมีแนวโน้มที่จะเกิด overflow เกิดขึ้นได้ เพราะ ตัวถูกดำเนินการมีค่าค่อนข้างเล็ก เมื่อนำมาคูณอาจจะสามารถทำให้จำนวนของ int เกิด overflow ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การแก้ปัญหาข้อที่หนึ่ง ซึ่งใช้ IA-32 ซึ่งใช้ mul เป็นคำสั่ง ที่ใช้ในการจัดสรรและเก็บข้อมูลของผลิตภัณฑ์นี้ไว้และจะต้องมีขนาดเป็น 2 เท่าของ operand ที่มีขนาดใหญ่ ตัวอย่างเช่น ผลิตภัณฑ์ของทั้งสองตัวนำมาคูณกันมีขนาด 8 บิตและ ให้แทนด้วย 16 บิต และ ผลิตภัณฑ์ของ 16 บิตที่มาคูณกันและให้แทนด้วย 32 bit

ตาราง C-style คำสั่ง mul จะถูกยอมรับด้วย 8,16,32 bit ของoperand และจะเก็บผลลัพธ์ไว้ใน 16,32 และ 64 bit

โปรแกรม 2.19 IA-32 unsigned multiplication [Intel 04]

```
1.if (OperandSize == 8) {
 2. AX = AL * SRC;
 3.else {
 4. if (OperandSize == 16) {
 5.  DX:AX = EAX * SRC;
 6. }
 7. else {  // OperandSize == 32
 8.  EDX:EAX = EAX * SRC;
 9. }
```

โปรแกรม 2.19 แสดงคำสั่ง assembly โดยใช้ Visual c++ ในการคอมไพเลอร์สำหรับ signed chae ,unsigned char และ unsigned int ที่ใช้ในการคูณ โดยใน 4 กรณีนี้ มี 2 operand ได้จากคำสั่งที่ใช้ imul จากคำสั่ง imul ก็สามารถใช้ signed และ unsigned int ได้ นั้นเอง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรม 2.20 Assembly code generated by Visual C++

```
sc_product =sc1 *sc2;
 1.movsx         eax, byte ptr [sc1]
 2.movsx         ecx, byte ptr [sc2]
 3.imul          eax, ecx
 4.mov           byte ptr [sc_product], al
uc_product =uc1 *uc2;
 5.movzx         eax, byte ptr [uc1]
 6.movzx         ecx, byte ptr [uc2]
 7.imul          eax, ecx
 8.mov           byte ptr [uc_product], al
si_product =si1 *si2;
ui_product =ui1 *ui2;
 9.mov           eax, dword ptr [ui1]
10.imul          eax, dword ptr [ui2]
```

Error Detection คำสั่งใน IA-32 ได้มีการนำชุดคำสั่งการ หารดังต่อไปนี้ div, divpd, divps, divsd, divss, fdiv, fdivp, fidiv และ idiv ซึ่งชุดของคำสั่ง div ที่ใช้ในการหาร (unsigned) integer มีค่าใน register เป็น ax, dx:ax, หรือ edx:eax(ผลหาร) โดย source จะถูกเก็บไว้ใน ax (ah:al), dx:ax, หรือ edx:eax ของ register คำสั่ง idiv จะมีประสิทธิภาพเหมือนกับค่าที่อยู่บน (signed) ผลลัพธ์ของคำสั่ง div/idiv จะขึ้นอยู่กับขนาดของตัวดำเนินการ(ผลหาร/ตัวหร) จากรูป 2.10 จะแสดงชุดคำสั่งสำหรับ sign (idiv)

Operand size	Dividend	Divisor	Quotient	Rem.	Quotient range
Word/byte	ax	r/m8	al	ah	-128 to +127
Doubleword/word	dx:ax	r/m16	ax	dx	-32,768 to +32,767
Quadword/doubleword	edx:eax	r/m32	eax	edx	-2^{31} to $2^{32}-1$

รูป 2.10 the idiv Instruction

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรม 2.21 แสดงคำสั่ง assembly ของ intel ที่ถูกสร้างโดย Visual c++ ใช้สำหรับ signed และ unsigned ในการหาร โดย signed ที่ใช้การหาร จะใช้ คำสั่ง idiv และ unsigned ที่ใช้ในการหารจะใช้คำสั่ง div เพราะตัวหารใน signed และ unsigned ในกรณีนี้จะใช้ค่า 32 bit ซึ่งผลหารจะถูกตีความเป็น quadword ในกรณีของ signed จะถูกจัดการโดยเพิ่มขนาดของ si_dividend ใน EAX โดยใช้วิธีขยาย sign และจัดเก็บผลลัพธ์ไว้ ใน register ของ edx:eax ใน กรณีของ unsigned ใน register EAX จะถูกลบโดยใช้คำสั่ง xor ก่อนจะเรียกคำสั่ง div เพื่อทำให้แน่ใจว่าไม่มีค่าใน register แล้ว

โปรแกรม 2.21 Assembly code generated by Visual C++

```
si_quotient =si_dividend /si_divisor;
 1.mov  eax, dword ptr [si_dividend]
 2.cdq
 3.idiv eax, dword ptr [si_divisor]
 4.mov  dword ptr [si_quotient], eax
ui_quotient =ui1_dividend /ui_divisor;
 5.mov eax, dword ptr [ui_dividend]
 6.xor edx, edx
 7.div eax, dword ptr [ui_divisor]
```

2.7 Mitigation Strategies

ช่องโหว่ของชนิดข้อมูล integer ทั้งหมดเป็นผลมาจากช่วงของชนิดข้อมูลของ integer ที่เกิดการความผิดพลาดขึ้น ยกตัวอย่างเช่น integer overflow เกิดขึ้นเมื่อมีการดำเนินงานสร้าง integer ที่อยู่นอกช่วงของชนิดข้อมูล integer นั้นและ Truncation errors เกิดขึ้นเมื่อค่าที่จะถูกเก็บไว้ในของตัวแปรที่เป็นประเภทที่มีขนาดเล็กเกินไปและ Sign errors เกิดเมื่อค่านั้นเป็นค่าลบและจะถูกบันทึกในรูปแบบการลงนามที่ไม่สามารถรองรับช่วงค่านั้นได้เพราะช่องโหว่ของ integer ทั้งหมดที่มีข้อผิดพลาดจะอยู่ในช่วงของชนิดของการตรวจสอบ ถ้านำไปใช้อย่างถูกต้องจะสามารถกำจัดช่องโหว่ของ integer ทั้งหมดได้โดย

กลยุทธ์เพื่อใช้ในการจัดการความเสี่ยงและลดช่องโหว่ต่างๆในการเขียนภาษา C และ C++ ดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.7.1 Range Checking

คุณอาจจะคาดหวังว่าภาระการตรวจสอบในภาษา C และ C++ ในเรื่องของช่วงของ integer นั้นจะเป็นหน้าที่ของโปรแกรมเมอร์เดียว ยกตัวอย่างเช่น การตรวจสอบว่าค่าของ integer นั้นอยู่ในช่วงที่เหมาะสมก่อนที่นำใช้กับ index ของ array จาก โปรแกรม 2.22 เป็นช่องโหว่ของ sign-error ตัวอย่างจะแสดงให้เห็นค่าของช่วง(integer ที่เป็นลบ) สามารถนำมาใช้เพื่อหลีกเลี่ยงการตรวจสอบขอบเขตบนของ array และก่อเกิด buffer overflow ได้

โปรแกรม 2.22 Implementation containing a sign error

```
1.#define BUFF_SIZE 10
2.int main(int argc, char*argv[]){
3. int len;
4. char buf[BUFF_SIZE];
5. len =atoi(argv[1]);
6. if (len < BUFF_SIZE){
7.  memcpy(buf, argv[2], len);
8. }
9.  else
10. printf("Too much data\n");
```

โดยเราสามารถใช้ Range Checking มาช่วยแก้ปัญหาที่เกิดขึ้นได้จาก โปรแกรม 2.23 มีการใช้ implicit type checking และ Explicit range check ผลของการทำ Implicit type check จะอยู่บนบรรทัด 3 โดย len เป็น unsigned integer โดยทั่วไปแล้วมันเป็นการดีที่เก็บเป็นชนิดข้อมูลแบบ unsigned integer สำหรับ indices และ ขนาด และ loop เพราะว่าไม่ควรที่จะให้มันเป็นค่าลบ บนบรรทัดที่ 7 เรียกใช้ฟังก์ชัน memcpy() จะถูกป้องกันโดยการทำ explicit range บนบรรทัดที่ 6 ใช้สำหรับทดสอบค่าทั้งสองว่าโดยการเช็คค่าของ len มากกว่า 0 และเช็คค่าของ len ว่าน้อยกว่าขนาดของ BUFF_SIZE ที่จองไว้หรือเปล่าเพื่อป้องกันการเกิด Overflow

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรม 2.23 Implementation containing a sign error

```
1. #define BUFF_SIZE 10
2. int main(int argc, char*argv[]){
 3. unsigned int len;
 4. char buf[BUFF_SIZE];
 5. len =atoi(argv[1]);
 6. if ((0 < len)&& (len < BUFF_SIZE)){
 7.  memcpy(buf, argv[2], len);
 8. }
 9. else
10.  printf("Too much data\n");
```

การใช้การเช็คทั้งสองแบบ implicit และ explicit อาจจะเป็นวิธีการที่มากพอแล้วแต่เราควรที่จะมีความระมัดระวังมากกว่านี้เช่น การเช็ค ค่าที่มาจากภายนอกควรมีการกำหนดค่าหรือสามารถบอกได้ว่าเป็น upper bound หรือ lower bound

2.7.2 Strong type

วิธีหนึ่งที่จะช่วยตรวจสอบชนิดของตัวแปรที่ดีคือการใช้ชนิดของตัวแปรที่เหมาะสม เช่น การใช้ชนิด unsigned สามารถที่จะบอกได้ว่าตัวแปรนั้นจะไม่มีค่าลบ เพื่อป้องกันไม่ให้เกิด overflow Strong type ควรที่จะใช้เพราะว่าจะทำให้คอมไพเลอร์มีประสิทธิภาพมากขึ้นในการระบุปัญหาต่างๆที่เกิดขึ้น

ตัวอย่าง ของการใช้ Strong type การประกาศตัวแปร integer ที่ใช้เก็บ อุณหภูมิ ของน้ำ โดยใช้ สเกลของ Fahrenheit

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรม 2.24 ตัวอย่าง ของการใช้ Strong type

```
unsigned char waterTemperature;

waterTemperature จะเป็น unsigned ที่มีค่า 8 bit ที่อยู่ในช่วง 1-255
unsigned char
```

เพียงพอที่จะใช้แทนค่าอุณหภูมิของน้ำซึ่งอยู่ในช่วงจาก 32 ดีกรี Fahrenheit(จุดเยือกแข็ง) ถึง 212 ดีกรีFahrenheit(จุดเดือด)ทำให้ป้องกันการเกิด overflow ได้

2.7.3 Abstract Data Type

หนึ่งในการแก้ปัญหาของชนิด Abstract Data ซึ่งในตัวแปร waterTemperature x ประกาศไว้ในเป็น private และไม่สามารถที่ใช้ค่านั้นได้โดยตรงจากผู้ใช้
ผู้ใช้ข้อมูล Abstraction นั้นสามารถจะเข้าถึงได้และอัพเดท หรือ กระทำการใดๆ กับค่านั้นด้วยการเรียกใช้ public method ซึ่งวิธีการนี้ต้องจัดเตรียมชนิดของตัวแปรที่ใช้เพื่อความปลอดภัยโดยต้องทำให้แน่ใจว่าค่าของ waterTemperature จะไม่สามารถออกจากค่าที่กำหนดไว้ได้ ถ้าหากมีการดำเนินการที่ถูกวิธีแล้วก็อาจจะเป็นไปได้ว่าเกิดข้อผิดพลาดของช่วง integer ขึ้นได้

2.7.4 Compiler Checks

Visual C++ .NET 2003 จะมีการเตือนค่าของ integer ที่ใช้ที่มีค่าเล็กเกินไป (C4244) เช่น การเตือนขั้นที่ 1 อาจจะบอกว่า ถ้า_int64 มีการใส่ค่าใน unsigned integer การเตือนขั้นที่ 3 และ 4 อาจเตือนด้วยข้อความว่า "ข้อมูลอาจจะสูญหายได้" ถ้ามีการแปลง integer ไปเป็นค่าที่เล็กกว่า สำหรับตัวอย่าง ตามที่มีการเตือนในขั้นที่ 4

โปรแกรม 2.25 การแปลง integer ไปเป็นค่าที่เล็กกว่า

```
int main(){
        int b =0, c =0;

        short a =b +c;    //C4244
    }
```

Visual C++ .NET 2003 จะมีการเช็คในขณะ runtime เพื่อจับ truncation error ของ integer ที่มีการใส่ค่าน้อยกว่าตัวแปรนั้นเป็นผลให้ค่านั้นสูญหายนั้นเอง /RTCc compiler จะมีการจับ flag error และจะสร้างผลการรายงานความผิดพลาดมาให้ Visual C++ จะการนำค่าของ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

runtime_check ซึ่งมันสามารถที่จะไม่ใช้ หรือ คืนค่าในการตั้งค่า /RTC แต่มันไม่สามารถนำค่า flag เพื่อใช้สำหรับจับ runtime error อื่นได้เช่น overflow

2.7.5 GCC Runtime Checks

gcc และ g++ compiler จะมีการนำค่า ftrapv compiler จะเป็นตัวเลือกในการจำกัดและสำหรับในการตรวจสอบ integer ที่ถูกยกเว้นในขณะ runtime แสดงไว้ในโปรแกรม 2.26 ระบบ runtime ของ gcc จะใช้การตรวจสอบ overflow โดยได้จากผลของการบวก ของ integer signed 16 bitและ บรรทัด 3 การบวกจะถูกดำเนินการและผลของการบวกถูกเปรียบเทียบกับตัวถูกดำเนินการว่าเกินค่าที่กำหนดไว้หรือเปล่าถ้าเกินก็จะเกิด error ขึ้น บรรทัด 5 จะแสดง abort () จะถูกเรียก ถ้า b ไม่เป็นค่าลบ และ w< a หรือ b เป็นค่าลบ และ w > a

โปรแกรม 2.26 Postcondition approach for overflow checking

```
1.Wtype__addvsi3 (Wtype a, Wtype b){
2. const Wtype w =a +b;
3. if (b >=0 ? w < a :w > a)
4.  abort ();
5. return w;
```

2.7.6 Safe Integer Operations

การดำเนินงานของ integer อาจจะส่งผลให้เกิดเงื่อนไขที่ผิดพลาดได้และอาจเกิดการสูญหายได้วิธีแรกที่ใช้ในการป้องกันช่องโหว่ integer ควรจะตรวจสอบช่วง integer

Explicitly Range check

Implicitly Range check

มันเป็นการยากที่จะรับประกันว่าได้ว่าปัจจัยต่างๆที่ไม่สามารถจัดการได้นั้นอาจจะนำมาซึ่งข้อผิดพลาดที่เกิดขึ้นกับโปรแกรมบางส่วนได้

วิธีป้องกันจะใช้ safe integer ไลบรารี่ สำหรับการดำเนินการทั้งหมดที่ทำใน integer

ตัวอย่าง ใน ภาษา C จะมี compatible library

ซึ่งเขียนโดย Michael Howard ของ Microsoft

สามารถตรวจจับเงื่อนไข integer overflow โดย IA-32 ที่ระบุไว้ใน กลไก

จากโปรแกรม 2.27 บรรทัดที่ 2 จะใช้ UAdd () ช่วยในตรวจสอบตามความเหมาะสมสำหรับเงื่อนไขที่ผิดพลาด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรม 2.27 C language compatible library solution

```
1.int main(int argc, char *argv[]){
 2. unsigned int total;
 3. if (UAdd(strlen(argv[1]), 1, &total)&&
          UAdd(total, strlen(argv[2]), &total)){
 4.  char *buff =(char *)malloc(total);
 5.  strcpy(buff, argv[1]);
 6.  strcat(buff, argv[2]);
 7. } else {
 8. abort();
 9. }
```

2.7.7 Safe Int Class

Safe Int ใน c++ จะใช้การเขียน class c++ template โดย David LeBlanc วิธีการดำเนินการที่จำเป็นและใช้ในการทดสอบค่าของตัวถูกดำเนินก่อนที่จะทำการดำเนินการเพื่อใช้ในการตรวจสอบข้อมูลผิดพลาดของข้อมูลจาก โปรแกรม 2.28 บรรทัดที่ 7 เป็นตัวแปร s1 และ s2 ที่ประกาศไว้เป็น SafeInt และ บรรทัดที่ 9 เมื่อทำการบวกกันก็จะเรียกใช้เวอร์ชันที่ปลอดภัยโดยสร้างผ่าน Class Safe Int

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรม 2.28 C language compatible library solution

```
1.//addition
2.SafeInt<T> operator +(SafeInt<T> rhs){
3.return SafeInt<T>(addition(m_int,rhs.Value()));
4.}
5.int main(int argc, char *const *argv){
6. try{
7.  SafeInt<unsigned long> s1(strlen(argv[1]));
8.  SafeInt<unsigned long> s2(strlen(argv[2]));
9.  char *buff =(char *)malloc(s1 +s2 +1);
10.  strcpy(buff, argv[1]);
11.  strcat(buff, argv[2]);
12.  }
13.  catch(SafeIntException err){
14.   abort();
```

แสดงการเปรียบเทียบเมื่อใช้ Safe Integer และ ไม่ใช้ Safe Integer

ตัวอย่าง ที่ใช้ Safe Integer จะสามารถจัดการค่าของ integer ที่มาจากแห่งที่ไม่น่าเชื่อถือ สำหรับตัวอย่างนี้คือ

1) ขนาดของโครงสร้าง

2) จำนวนของโครงสร้างในการจองพื้นที่

จากตัวอย่าง จะเห็นว่าบรรทัดที่ 3. ซึ่งการคูณนี่จำเป็นที่จะต้องกำหนดขนาดของหน่วยความจำ และบรรทัดที่ 4. การคูณอาจจะทำให้เกิด overflow กับ integer ได้และทำให้เกิดช่องโหว่ buffer overflow ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรม 2.29 การใช้ Safe Integer

```
void* CreateStructs(int StructSize, int HowMany){
        SafeInt<unsigned long> s(StructSize);
          s *= HowMany;
          return malloc(s.Value());
```

ตัวอย่าง ไม่ใช้ Safe Integer

ถ้าไม่ใช้ safe integer เมื่อเกิด overflow จะอาจทำให้เกิดปัญหา tight loop ได้ดังนี้ตัวอย่างต่อไปนี้

โปรแกรม 2.30 ไม่ใช้ Safe Integer

```
void foo(){
 char a[INT_MAX];
       int i;
     for (i =0; i < INT_MAX; i++)
         a[i]='\0';
```

2.7.8 Testing

การทดสอบช่องโหว่ของ integer ควรจะรวมเงื่อนไขและขอบเขตสำหรับตัวแปรทั้งหมดของ integer เช่น ถ้าช่วงของชนิดที่จะถูกเช็คอยู่ใน code ที่เพิ่มเข้ามา จะถือว่าการทดสอบนั้นถูกต้องสำหรับขอบเขตบนและล่างถ้าการทดสอบไม่ได้อยู่ในขอบเขตค่าต่ำสุดและค่าสูงสุดของ integer สำหรับขนาดต่างๆที่ใช้จะใช้ white box testing ในการกำหนดค่าและชนิดของตัวแปร integer ถ้าซอร์สโค้ดไม่สามารถเรียกใช้การทดสอบได้ให้ทดสอบจากค่าสูงสุดและค่าต่ำสุดที่แตกต่างกันสำหรับแต่ละประเภท

2.7.9 Source Code Audit

ซอร์สโค้ดควรได้รับการตรวจสอบหรือการตรวจสอบข้อผิดพลาดที่จำเป็นของช่วง integer ดังต่อไปนี้ ช่วงของ integer ควรจะตรวจสอบความถูกต้องเสมอและค่าที่ป้อนเข้ามาจะถูกจำกัดให้อยู่ในช่วงที่ถูกต้องตามงานที่ใช้ integer ที่ไม่ต้องใช้ค่าลบควรจะประกาศเป็น unsigned และ ตรวจสอบขอบเขตบนและขอบเขตล่าง การดำเนินงานกับ integer ที่มาจากแหล่งที่ไม่น่าเชื่อถือควรจะดำเนินการโดยใช้ safe integer library

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.8 Formatted Output Vulnerabilities

2.8.1 Formatted Output

Formatted Output จะประกอบด้วย format string และค่าของจำนวนของตัวแปรของอาร์กิวเมนต์ Format string ก็เป็นรูปแบบที่ใช้กำหนดลักษณะการรับข้อมูล โดยการควบคุมข้อมูลของ format string สามารถควบคุมการส่งออกค่าโดยผู้ใช้ได้ เพราะฟังก์ชันทั้งหมด ของ formatted output มีความสามารถที่ช่วยให้ผู้ใช้สามารถละเมิดความปลอดภัยที่มีอยู่ได้นั้นเอง

2.8.2 Variadic Functions

Variadic Functions จะใช้ในระบบ UNIX ของระบบ V หรือ ANSI C ซึ่งทั้งสองต้องติดต่อระหว่างนักพัฒนา และผู้ใช้ฟังก์ชัน Variadic ให้ไม่ถูกละเมิดโดยผู้ใช้

2.8.3 ANSI C Standard Arguments

ANSI C Standard Arguments (ยังสามารถเรียกว่า stdargs) ฟังก์ชัน variadic มีการประกาศใช้รายการพารามิเตอร์ partial ที่ใช้แสดงรายชื่อที่ติดตามโดย มีบางส่วนตามเครื่องหมายจุดไข่ปลาสำหรับตัวอย่างนี้ ที่ใช้ฟังก์ชัน variadic average() แสดงใน โปรแกรม 2.31

โปรแกรม 2.31 mentation of average() function using stdargs

```
1.int average(int first, ...){
 2. int count =0, sum =0, i =first;
 3. va_list marker;

 4. va_start(marker, first);
 5. while (i !=-1){
 6.  sum +=i;
 7.  count++;
 8.  i =va_arg(marker, int);
 9. }
10. va_end(marker);
11. return(sum ? (sum /count):0);
```

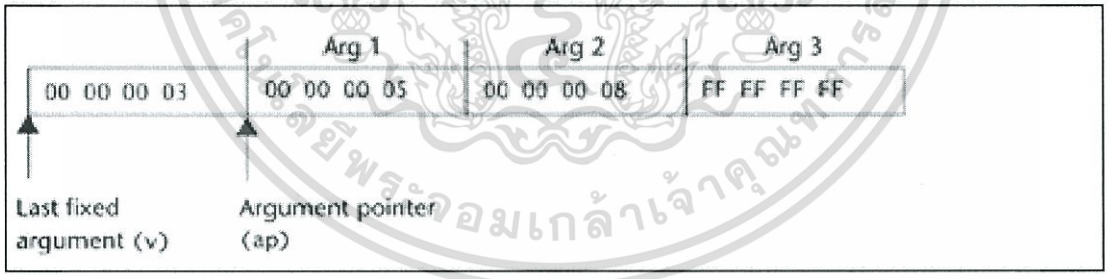
ฟังก์ชันที่มีจำนวนของตัวแปรของ argument ที่ใช้โดยการระบุจำนวนที่ต้องการของการเรียกใช้ฟังก์ชัน average (3,5,8,-1)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ANSI C จะมี va_start(),va_arg() และ va_end() ที่ใช้เป็นมาโคร variadic ฟังก์ชันซึ่งมาโครนี้จะถูกระบุไว้ใน stdarg.h ในไฟล์ include ซึ่งการดำเนินการทั้งหมดบนชนิดข้อมูล va_list และรายการอาร์กิวเมนต์ที่ประกาศและใช้เป็นชนิด va_list จาก โปรแกรม 2.32 มีการใช้ชนิดข้อมูลเป็น va_list และมี va_strat,va_arg(), และ va_end() มาโครที่ระบุโดย Visual C++ .NET compiler สำหรับ IA-32 ชนิดที่ระบุไว้ใน va_list ที่เป็น character pointer

โปรแกรม 2.32 Sample definition of variable argument macros

```
#define _ADDRESSOF(v)(&(v))

#define _INTSIZEOF(n)\

((sizeof(n)+sizeof(int)-1)& ~(sizeof(int)-1))


typedef char *va_list;


#define va_start(ap,v)
(ap=(va_list)_ADDRESSOF(v)+_INTSIZEOF(v))

#define va_arg(ap,t)(*(t *)((ap+=_INTSIZEOF(t))-_INTSIZEOF(t)))

#define va_end(ap)(ap =(va_list)0)
```



รูป 2.11 Type va_list as character pointer

รูป 2.11 แสดง va_list ที่เป็น character pointer ที่อยู่บนลำดับของ Stack เมื่อมี average(3,5,8,-1) character pointer ที่ระบุไว้โดย va_start() จะถูกอ้างถึงตามพารามิเตอร์ของอาร์กิวเมอร์ที่แก้ไขล่าสุด มาโคร va_start() ที่ใช้เพิ่มขนาดของตำแหน่งอาร์กิวเมนต์สุดท้ายที่ถูกแก้ไขเมื่อ va_start() คืนค่ามาและ va_list point ที่เป็นตำแหน่งแรกของที่จะใช้เลือกอาร์กิวเมนต์ การ Passing Argument และการ Convention Nameing มีการเรียก convention ที่จะถูก support โดย Visual C++ มีดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1) C Calling Convention (cdecl) - เป็นแบบที่ compiler ปัจจุบันใช้ โดยฟังก์ชันที่เรียกจะทำ หน้าที่ clear Stack เช่นการเรียกsome_args(1, 2, 3) จะได้ Assembly ของฟังก์ชัน ที่เรียกคือ

โปรแกรม 2.33 C Calling Convention (cdecl)

```
push $3
push $2
push $1
call some_args
add $12,%esp
```

2) Standard Convention (stdcall) การเรียกแบบนี้ Microsoft เป็นคนคิด และใช้ใน dll ของ Microsoft เอง ถ้าใครเคยเขียนโปรแกรมโดยใช้ WIN32 API คงจะเคยเห็น WINAPI หน้า function ซึ่งถ้าไล่ดูใน header file ของ WIN32 API ก็จะเห็นว่า define เป็น _stdcall การเรียกแบบนี้ต่างจากแบบแรกคือ ฟังก์ชันที่ถูกเรียกจะทำหน้าที่ clear stack โดยใช้คำสั่ง Assembly "RET n" เช่นการเรียก some_args(1, 2, 3) จะได้ Assembly ของฟังก์ชัน ที่เรียกคือ

โปรแกรม 2.34 Standard Convention (stdcall)

```
push $3
push $2
push $1
call some_args
```

3) fastcall แบบนี้จะคล้ายแบบ "Standard Convention" ต่างกันตรงที่ argument ตัวแรกจะเก็บไว้ใน ECX และตัวที่สองเก็บไว้ใน EDX ส่วนที่เหลือ push ลง Stack เหมือนเดิม

2.8.4 Formatted Output Functions

Formatted output ฟังก์ชันจะถูกระบุโดยพื้นฐานของ C99 โดยมีดังนี้

1) fprintf() จะเขียน output ไป stream โดยขึ้นอยู่กับ format string
2) printf() จะเทียบได้กับ fprint() ยกเว้น printf() นั้น output ของ stream เป็น stdout

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3) print() จะเทียบได้กับ fprint ยกเว้น output ที่ถูกเขียนไว้บน array มากกว่าที่จะอยู่บน stream

4) snprintf() จะเทียบได้กับ 40print() ยกเว้น ค่าต่ำสุดของจำนวนตัวอักษร n ถึงค่าที่เขียนระบุไว้ เช่น ถ้า n ไม่เท่ากับ 0 output ของตัวอักษร ก็จะอยู่ไม่เกิน n-1

5) vfprintf(), vprintf(), vsprintf(), และ vsnprintf() ซึ่งเทียบได้กับ fprintf(), printf(),print() และ snprintf() กับ รายการอาร์กิวเมนต์จะถูกแทนโดยชนิดของอาร์กิวเมนต์ที่เป็น va_list ซึ่ง ฟังก์ชันเหล่านี้จะนำมาใช้ประโยชน์เมื่อรายการของอาร์กิวเมนต์นั้นถูกกำหนดที่ใน runtime ใหม่

2.8.5 Format Strings

Format string คือ รูปแบบที่ใช้กำหนดลักษณะการรับข้อมูล

ตาราง 2.1 Format String

format code	รูปแบบ
%d	สำหรับการแสดงผลตัวเลขจำนวนเต็ม (int, short, unsigned short, long, unsigned long
%u	สำหรับการแสดงผลตัวเลขจำนวนเต็มบวก (unsigned short, unsigned long
%o	สำหรับการแสดงผลออกมาในรูปแบบของเลขฐานแปด
%x	สำหรับการแสดงผลออกมาในรูปแบบของเลขฐานสิบหก
%f	สำหรับการแสดงผลตัวเลขทศนิยม (float, double, long double)
%e	สำหรับการแสดงผลตัวเลขทศนิยมออกมาในรูปแบบของ E (หรือ e) ยกกำลัง (float, double, long double)
%c	สำหรับการแสดงผลอักขระ 1 ตัว (char)
%s	สำหรับการแสดงผลข้อความ (อักขระมากกว่า 1 ตัว)
%p	สำหรับการแสดงผลตัวชี้ตำแหน่ง (pointer)

2.8.6 Exploiting Formatted Output Functions

ฟังก์ชัน Formatted output จะเขียนไปเป็น array character (สำหรับตัวอย่าง sprintf()) สมมุติว่า buffer มีความยาวมาก ซึ่งทำให้เกิดความเสี่ยงที่อาจทำให้เกิด buffer overflow รูปที่ 2.60 แสดงช่องโหว่ของ buffer overflow ที่ใช้ sprintf มันจะถูกแปลงระบุเป็น %s ด้วย (ซึ่งอาจจะเกิดอันตรายได้) สตริงจากผู้ใช้ สตริงใดๆ ยาวกว่า 495 ไบต์ ผลลัพธ์จะอยู่ นอกขอบเขตของการเขียน (512 ไบต์-16ไบต์-null อีก 1 ไบต์)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรม 2.35 Formatted output function susceptible to buffer overflow

```
1.char buffer[512];
2.sprintf(buffer, "Wrong command:%s\n", user);
```

โปรแกรม 2.35 sprintf() ไม่สามารถที่จะเรียกใช้ได้โดยตรงเพราะ % .400s จะถูกแปลงและจำกัดจำนวนไบต์ที่ใช้เขียนไป 400 และ บนบรรทัดที่ 4 สามารถใช้ในการโจมตีทางอ้อมได้ผ่าน sprint() ที่ใช้โดยการร้องขอจากผู้ใช้ซึ่งได้ค่าจากการทำ shellcode%497d\x3c\xd3\xff\xbf<nops><shellcode> sprintf() ที่เรียกใช้บนบรรทัดที่ 3 จะเพิ่มสตริงลงไปใน buffer ซึ่ง buffer array จะถูกส่งค่าผ่านไปใน sprintf() บนบรรทัดที่ 4 ซึ่งเป็น format string อาร์กิวเมนต์ ซึ่ง %497d จะถูกกำหนดในรูปแบบคำสั่ง sprintf() ที่ถูกอ่านอาร์กิวเมนต์จาก Stack และเขียนตัวอักษร 497 ลงบน buffer

โปรแกรม 2.36 Stretchable buffer

```
1.char outbuf[512];
2.char buffer[512];
3.sprintf(
      buffer,
  "ERR Wrong command:%.400s",
      user
 );
```

2.8.7 Output Streams

ฟังก์ชัน Formatted output นี่จะถูกเขียนไปใน stream แทนไฟล์(เช่น printf()) นอกจากนี้ยังมีความเสี่ยงของ format string จากโปรแกรม 2.37 มีช่องโหว่ของ format string ถ้าอาร์กิวเมนต์ของผู้ใช้สามารถควบคุมได้อย่างเต็มที่หรือบางส่วนโดยผู้ใช้โปรแกรมสามารถใช้ประโยชน์จากข้อผิดพลาดนี่ในดูเนื้อหาใน stack ดูขนาดของหน่วยความจำหรือเขียนทับหน่วยความจำได้นั้นเอง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรม 2.37 Exploitable format string vulnerability

```
1. int func(char *user) {
2.    printf(user);
```

2.8.8 Crashing a Program

ช่องโหว่ของ format string จะถูกค้นพบเมื่อโปรแกรมเกิดปัญหาสำหรับระบบ Unix ก็เข้าใช้แบบ invalid pointer อาจจะทำให้เกิดสัญญาณ SIGSEGV กับกระบวนการได้ เว้นแต่จะจับและจัดการกับโปรแกรมที่ผิดปกติและทำการ dump core ได้ การเข้าถึงแบบ invalid pointer หรือ unmapped address read สามารถที่เกิดจากการเรียกผ่าน formatted output :

โปรแกรม 2.38 unmapped address read

```
printf("%s%s%s%s%s%s%s%s%s%s%s");
```

การแปลงที่ %s ที่แสดงในหน่วยความจำในที่อยู่ในอาร์กิวเมนต์ที่สอดคล้องกับที่อยู่บน Stack เพราะ ไม่มี String อาร์กิวเมนต์ที่จัดมาในตัวอย่าง printf() อ่านตำแหน่งในหน่วยความจำจาก Stack จนกระทั่ง format String หมด หรือ เกิด invalid pointer หรือ พบ unmapped address

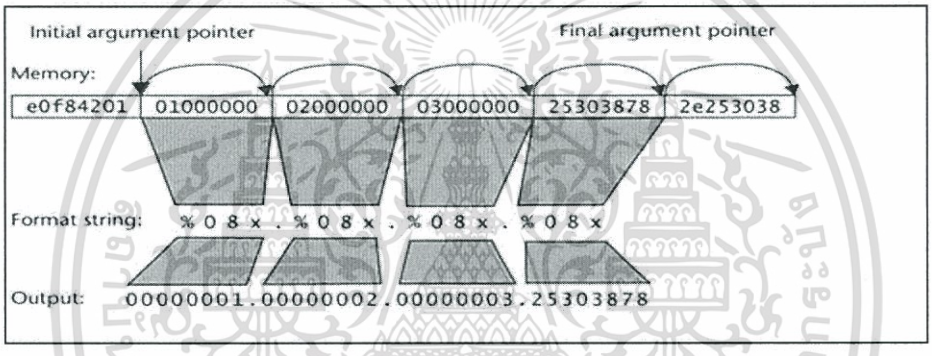
2.8.9 Viewing Stack Content

ผู้โจมตีสามารถใช้ช่องโหว่ฟังก์ชัน formatted output ตรวจสอบข้อมูลใน memory ได้ โปรแกรม 2.39 แสดงการ disassembled ฟังก์ชัน printf() บนบรรทัดที่ 2 อาร์กิวเมนต์ต่างๆจะถูกใส่ลงบน Stack ที่อยู่ใน reverse oreder และ บรรทัดที่ 5 อาร์กิวเมนต์ต่างๆที่อยู่ใหน่วยความจำที่ปรากฏอยู่ในลำดับเดียวกันจะเรียกใช้ printf ()

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรม 2.39 Disassembled printf() call

```
1. char outbuf[512];
2. char buffer[512];
3. sprintf(
     buffer,
     "ERR Wrong command: %.400s",
     user
   );
4. sprintf(outbuf, buffer);
```



รูป 2.12 Viewing the contents of the stack

2.8.10 Overwriting Memory

ฟังก์ชัน Formatted output นั้นจะมีการส่งค่าออกไปที่มีอันตราย เพราะ โปรเมอร์ส่วนใหญ่นั้นไม่รู้จักความสามารถของ Formatted output (เช่น Formatted output สามารถเขียนค่าจำนวนเต็มไปยังที่อยู่ที่ระบุไว้โดยใช้ตัวระบุ %n ได้) บนแพลตฟอร์มของจำนวนเต็มและมีขนาดเท่ากัน (เช่น IA-32) ความสามารถในการเขียน integer ไปยังที่อยู่โดยสามารถนำมาใช้เพื่อรันโค้ดบนระบบที่ถูกบุกรุกได้%n จะถูกสร้างขึ้นมาเพื่อช่วยจัดรูปแบบของสตริงเอาท์พุท ซึ่งมันจะเขียนจำนวนของตัวอักษรออกไปที่ตำแหน่งของ integer ที่จัดไว้ในอาร์กิวเมนต์ สำหรับตัวอย่างนี้หลังจากรันโค้ดต่อไปนี้

โปรแกรม 2.40 Overwriting Memory

```
int i;
printf("hello%n\n", (int *)&i);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ซึ่งค่า i ที่ถูกแทนค่าเป็น 5 เพราะ ตัวอักษรที่ถูกเขียนมี 5 ตัว (h-e-l-l-o) ก่อนที่จะแปลง %n ระบุไว้ ผู้โจมตีสามารถที่จะเขียนค่าของจำนวนเต็มไปยังตำแหน่งที่ต้องการได้ โดยอาศัยประโยชน์จากข้อบกพร่องของการรักษาความปลอดภัย เราสามารถระบุตำแหน่งโดยใช้เทคนิค examine memory ที่ใช้สำหรับระบุตำแหน่ง ดังตัวอย่างต่อไปนี้

โปรแกรม 2.41 Examine memory

```
printf("\xdc\xf5\x42\x01%08x%08x%08x%n");
```

การเขียนค่าของ integer พิจารณาจำนวนของตัวอักษรที่ตำแหน่ง 0x0142f5dc ในตัวอย่างนี้ ค่าที่เขียน(28) เทียบได้กับฐานสิบหก 8 ตัวบวกกับสี่ไบต์ จำนวนของตัวอักษรที่ถูกเขียนโดยฟังก์ชัน format จะขึ้นอยู่กับ format string ถ้าผู้โจมตีสามารถควบคุม format string ได้ ผู้โจมตีการสามารถที่จะควบคุมจำนวนของตัวอักษรที่ใช้เขียนได้ จากโปรแกรม 2.42 แสดงโค้ดที่ใช้ในการเขียน address ของที่ตั้งของหน่วยความจำที่เฉพาะเจาะจง นี้จะสร้างรูปแบบของสตริงในรูปแบบต่อไปนี้

% width u%n% width u%n% width u%n% width u%n

โปรแกรม 2.42 Exploit code to write an address

```
1.static unsigned int already_written, width_field;
2.static unsigned int write_byte;
3.static char buffer[256];
4.already_written =506;
//first byte
 5.write_byte =0x3C8;
 6.already_written %=0x100;
7.width_field =(write_byte -already_written)%0x100;
 8.if (width_field < 10)width_field +=0x100;
 9.sprintf(buffer, "%%%du%%n", width_field);
10.strcat(format, buffer);
//second byte
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรม 2.43 Exploit code to write an address(ต่อ)

```
12.already_written +=width_field;
13.already_written %=0x100;
14.width_field =(write_byte -already_written)%0x100;
15.if (width_field < 10)width_field +=0x100;
16.sprintf(buffer, "%%%du%%n", width_field);
17.strcat(format, buffer);
//third byte
18.write_byte =0x442;
19.already_written +=width_field;
20.already_written %=0x100;
21.width_field =(write_byte -already_written)%0x100;
22.if (width_field < 10)width_field +=0x100;
23.sprintf(buffer, "%%%du%%n", width_field);
24.strcat(format, buffer);
//fourth byte
```

โปรแกรม 2.44 Exploit code to write an address

```
26.already_written +=width_field;
27.already_written %=0x100;
28.width_field =(write_byte -already_written)%0x100;
29.if (width_field < 10)width_field +=0x100;
30.sprintf(buffer, "%%%du%%n", width_field);
31.strcat(format, buffer);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.9 Mitigation Strategies

2.9.1 Dynamic Use of Static Content

ข้อเสนอแนะในการขจัดรูปแบบของช่องโหว่ format string คือการไม่อนุญาตให้ใช้สตริงในรูปแบบไดนามิก ถ้า format string อยู่ในรูปแบบสแตติก ช่องโหว่ของ format string ก็จะไม่เกิดขึ้น (ยกเว้นในกรณีของหน่วยความจำเกิดล้นที่มาจากแถวอักขระไม่เพียงพอ) การแก้ปัญหานี้มันเป็นไปไม่ได้เพราะ สตริงในรูปแบบไดนามิกมีการใช้อย่างแพร่หลาย

ทางเลือกในการใช้กลยุทธ์แบบไดนามิกคือการใช้แบบ ไดนามิกโดยใช้ content ของ สแตกติก รูปที่ 2.12 แสดงให้เห็นว่าเป็นโปรแกรมอย่างง่ายๆที่คูณอาร์กิวเมนต์แรกโดยอาร์กิวเมนต์ที่สอง โปรแกรมยังใช้อาร์กิวเมนต์ที่สามที่สั่งให้โปรแกรมเกี่ยวกับวิธีการจัดรูปแบบการแสดงผล ถ้าอาร์กิวเมนต์ที่สามเป็นฐานสิบหกสตริงผลิตภัณฑ์ที่มีการแสดงในรูปแบบเลขฐานสิบหกโดยใช้ตัวระบุ% การแปลง x มิฉะนั้นก็จะแสดงเป็นตัวเลขทศนิยมโดยใช้ % และระบุ d ในการแปลง

โปรแกรม 2.45 Dynamic format strings

```
1.#include <stdio.h>
 2.#include <string.h>
 3.int main(int argc, char *argv[]){
 4. int x, y;
 5. static char format[256]="%d *%d =";
 6. x =atoi(argv[1]);
 7. y =atoi(argv[2]);
 8. if(strcmp(argv[3], "hex")==0){
 9.  strcat(format, "0x%x\n");
10. }
11. else {
12.  strcat(format, "%d\n");
13. }
14. printf(format, x, y, x *y);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.9.2 Restricting Bytes Written

เมื่อนำไปใช้ในรูปแบบของฟังก์ชันการส่งออกจะมีความอ่อนไหวต่อ format string และจะเกิด ช่องโหว่ buffer overflow ซึ่ง buffer overflow สามารถป้องกันได้โดยการจำกัด จำนวนไบต์ที่จะเขียนโดยฟังก์ชันเหล่านี้

โปรแกรม 2.46 Restricting Bytes Written

```
sprintf(buffer, "Wrong command:%s\n", user);

            try using

                        sprintf(buffer, "Wrong command:
%.495s\n", user);
```

2.9.3 Lexical Analysis

pscan [DeKok 00] เป็นเครื่องมือในการวิเคราะห์คำศัพท์ที่มาจากการสแกนโค้ดที่เป็นช่องโหว่ของ format string ผลของการสแกนโดยการค้นหาสำหรับฟังก์ชันในการส่งออกและการใช้กฎต่อไปนี้

ถ้าพารามิเตอร์ล่าสุดของฟังก์ชั่นที่เป็น format string และ format string ไม่เป็นแบบสแตติก แล้วให้แสดงออกมา

Libsafe

Libsafe รุ่น 2.0 จะป้องกันไม่ให้ช่องโหว่ของ format string ที่พยายามเขียนกลับมาบน stack และถ้าเกิดการโจมตีดังกล่าว Libsafe ก็จะบันทึกการเตือนและยุติกระบวนการเป้าหมายนั้นเอง

2.9.4 Staticd Binary Analysis

เป็นไปได้ที่จะค้นพบช่องโหว่ของ format string โดยการตรวจสอบจาก ภาพไบนารี โดยใช้เกณฑ์ต่อไปนี้

1) มีการแก้ไข stack ที่มีขนาดเล็กกว่าค่าต่ำสุดหรือเปล่า
2) เป็นตัวแปร format string หรือ ค่าคงที่ ยกตัวอย่างเช่น printf() ฟังก์ชั่นเมื่อใช้อย่างถูกต้องจะต้องยอมรับอย่างน้อยสองพารามิเตอร์ และ format string หาก printf () จะมีการถูกเรียก เพียงหนึ่งอาร์กิวเมนต์และนี้อาจเป็นตัวแปรที่ปีนตัวแทนของช่องโหว่ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จำนวนของอาร์กิวเมนต์ที่ส่งผ่านไปยังฟังก์ชันการส่งออกในรูปแบบที่สามารถกำหนดได้โดยการตรวจสอบการแก้ไข ในตัวอย่างต่อไปนี้จะเห็นได้ชัดว่ามีเพียงหนึ่งอาร์กิวเมนต์ถูกส่งผ่านไป printf () ฟังก์ชันเพราะการแก้ไข Stack เพียงแค่สี่ไบต์

โปรแกรม 2.47 Assembly printf() in Stack

```
 lea eax,[ebp+10h]
push eax
call printf
 add esp, 4
```

นอกจากนี้ยังเป็นไปได้ที่จะตรวจสอบว่าอาร์กิวเมนต์โดยการโหลด EAX ที่เป็นค่าคงที่ หรือตัวแปรโดยการตรวจสอบโค้ดของ assembly ซึ่งเครื่องมือนี้มีเพื่อช่วยตรวจสอบว่าอาร์กิวเมนต์ของตัวแปรสามารถได้รับอิทธิพลจากผู้ใช้ถึงแม้ว่าขั้นตอนนี้มีความซับซ้อนมากขึ้นนั่นเอง เทคนิคการวิเคราะห์แบบไบนารีจะยังคงสามารถนำมาใช้โดยนักพัฒนาหรือผู้ทดสอบการประกันคุณภาพของการค้นพบช่องโหว่โดยผู้ใช้ในการประเมินไม่ว่าจะเป็นผลิตภัณฑ์ที่มีความปลอดภัยหรือโดยการโจมตีที่เพื่อจะค้นพบช่องโหว่

2.10 File I/O Vulnerabilities

2.10.1 Race Conditions

คือสภาวะที่โปรเซส 2 โปรเซส ที่ต้องการใช้ทรัพยากรที่แชร์ไว้พร้อมกัน ในเวลาเดียวกัน ทำให้ผลลัพธ์อาจจะเกิดการผิดพลาดขึ้นได้ขึ้นอยู่กับว่าโปรเซสใดทำงานเสร็จก่อน

2.10.2 Mutual Exclusion and Deadlock

เป็นการแก้ปัญหา Race Condition คือ ป้องกันไม่ให้โปรเซสอื่นเข้ามาครองครองทรัพยากรได้ในขณะที่โปรเซสอื่นครอบครองอยู่เรียกว่า Mutual Exclusion คือการทำทีละprocess

Deadlock คือ เมื่อโปรเซสต้องการใช้ทรัพยากร และร้องขอทรัพยากรจากระบบ แต่ทรัพยากรที่โปรเซสต้องการใช้ไม่ว่าง เนื่องจากโปรเซสอื่นกำลังใช้งานอยู่ ทำให้โปรเซสนั้นๆ ต้องรอคอย และเป็นการรอคอยที่ไม่มีที่สิ้นสุด (โปรเซสไม่มีโอกาสได้รับการจัดสรรทรัพยากร)

2.10.3 Time of Check, Time of Use

ตัวอย่างนี้จะ พิจารณาโปรแกรมลินุกซ์ ที่แสดงในโปรแกรม 2.48 access() ฟังก์ชั่นที่เรียกใช้ในบรรทัดที่ 5 จะเป็นการตรวจสอบการมีอยู่ไฟล์และตรวจสอบว่าสามารถเขียนไฟล์ได้หรือเปล่า ถ้าเงื่อนไขเหล่านี้ถูกแฟ้มก็จะถูกเปิดออกสำหรับการเขียนที่แสดงไว้บนบรรทัดที่ 7

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรม 2.48 Code with TOCTOU condition on file open

```
1.#include <stdio.h>
2.#include <unistd.h>

3.int main(int argc, char *argv[]){
4. FILE *fd;

5. if (access("/some_file", W_OK)==0){
6.  printf("access granted.\n");
7.  fd=fopen("/some_file", "wb+");
8.  /*write to the file */
9.  fclose(fd);
10. }
  ...
11. return 0;
```

2.10.4 Files as Locks and File Locking

Race conditions จะเป็นกระบวนการที่เป็นอิสระและไม่สามารถแก้ไขโดยการ synchronization เพราะ กระบวนการนี่ไม่ได้ใช้ร่วมกันกับ global data (เช่น ตัวแปร mutex) ชนิดนี้จะใช้กระแสควบคุมพร้อมทั้งยังสามารถทำข้อมูลให้ตรงกันโดยใช้ lock ไฟล์ได้ ตัวอย่าง มีสองฟังก์ชันที่ใช้กลไกการล็อคไฟล์ของลินุกซ์ การเรียกร้องให้ล็อค A() จะใช้ในการได้รับการล็อคและล็อคจะถูกปล่อยออกโดยการเรียกใช้ unlock ()

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรม 2.49 Simple file locking in Linux

```
 1. int lock(char *fn) {
 2.   int fd;
 3.   int sleep_time = 100;
 4.   while (((fd=open(fn, O_WRONLY | O_EXCL |
        O_CREAT, 0)) == -1) && errno == EEXIST) {
 5.     usleep(sleep_time);
 6.     sleep_time *= 2;
 7.     if (sleep_time > MAX_SLEEP)
 8.       sleep_time = MAX_SLEEP;
 9.    }
10.   return fd;
11. }

12. void unlock(char *fn) {
13.   if (unlink(fn) == -1) {
14.     err(1, "file unlock");
15.   }
16. }add esp, 4
```

ซึ่งทั้ง lock() และ unlock() จะถูกส่งผ่านไปยังชื่อของแฟ้มที่ทำหน้าที่เป็น shared lock ของ object ซึ่งจะใช้ในกระบวนการ sharing และไดเร็กทอรี่ที่สามารถใช้ shared (directory/tmp เป็นที่นิยมที่ใช้เพื่อการนี้) แฟ้มล็อคจะถูกใช้เป็นพร็อกซี่สำหรับล็อคไฟล์ หากไฟล์ ที่มีอยู่ถูกล็อค หรือถูกจับและ ถ้าไฟล์ไม่ได้ล็อคอยู่จะถูกปล่อยออกมานั้นเอง

2.10.5 Symbolic Linking Exploits

พบมากที่สุดช่องโหว่ของ race-related file ที่เกี่ยวข้องกับกลไกของการ symbolic link ใน UNIX ซึ่ง symbolic link เป็นรายการของไดเรกทอรี่ที่ใช้อ้างอิงแฟ้มเป้าหมายหรือไดเร็กทอรี่ ช่องโหว่ symlink จะเกี่ยวกับการเขียนโปรแกรมเพื่อใช้อ้างอิงชื่อไฟล์ที่จะรวมใน symlink และ ช่องโหว่ที่พบมากที่สุด symlink จะเกี่ยวข้องกับเงื่อนไขของ TOCTOU ซึ่ง ช่องโหว่ TOCTOU อาจจะมีการเรียกใช้ access() ฟังก์ชัน ตามด้วย fopen() ที่แสดงใน โปรแกรม 2.50 แสดงให้เห็นถึง TOCTOU ที่สามารถที่จะนำไปสู่ช่องโหว่ symlink ซึ่งโค้ดนี้ stats /some_dir/some_file และ การเปิดไฟล์สำหรับอ่านถ้ามันไม่ได้มีขนาดใหญ่เกินไป ซึ่งการตรวจสอบ TOCTOU จะเกิดขึ้นกับการเรียก stat() บนบรรทัดที่ 1 และ TOCTOU จะใช้เรียกฟังก์ชัน fopen() บนบรรทัดที่ 8

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรม 2.50 Code with TOCTOU vulnerability involving stat()

```
1.if (stat("/some_dir/some_file", &statbuf)==-  1){
 2. err(1, "stat");
 3.}
 4.if (statbuf.st_size >=MAX_FILE_SIZE){
 5. err(2, "file size");
 6.}
 7.
 8.if((fd=open("/some_dir/some_file", O_RDONLY))      ==-1){
 9. err(3, "open -/some_dir/some_file");
10.}
11.//process file  add esp, 4
```

2.10.6 Temporary File Open Exploits

โปรแกรมมักจะมีการใช้งาน temporary file ปัญหาหลักที่มีกับ temporary file คือการที่จะไม่สามารถมั่นใจว่าในการตั้งชื่อไฟล์ให้ไม่ซ้ำกัน temporary file มีความเสี่ยงโดยเฉพาะอย่างยิ่งเมื่อสร้างขึ้นในไดเรกทอรี่ที่ผู้โจมตีสามารถเข้าถึงได้ ปัญหานี้เป็นปัญหาในระบบ UNIX เนื่องจากใช้ /tmp บ่อย ซึ่งเป็นไดเรกทอรี่ที่ต้องการสำหรับ temporary file

ตัวอย่างที่ง่ายที่สุดของช่องโหว่ดังกล่าวที่ไม่เกี่ยวข้องกับ race condition พิจารณาคำสั่งต่อไปนี้สำหรับการเปิดไฟล์ที่ส่งออกของไฟล์ใหม่

โปรแกรม 2.51 Temporary File Open Exploits

```
int fd =open("/tmp/some_file",O_WRONLY | O_CREAT |
O_TRUNC, 0600);
```

ถ้า /tmp/some_file มีอยู่แล้วในเวลาที่มีการดำเนินการคำสั่งแฟ้มจะถูกเปิดออกและถูกตัดทอน นอกจากนี้หาก /tmp/some_file เป็น symbolic link แล้วไฟล์เป้าหมายทีจะใช้อ้างอิงโดยการเชื่องโยงเป็นแบบตัดทอน ผู้โจมตีต้องการที่จะทำคือการสร้างการเชื่องโยงแบบ symbolic link โดยเรียก / tmp / some_file ก่อนที่จะรันคำสั่ง ซึ่งช่องโหว่จะรุนแรงมากขึ้นถ้าสามารถยกกระดับสิทธิ์ได้นั้นเอง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กลยุทธ์การบรรเทาผลกระทบที่ดีที่สุดสำหรับ UNIX ของช่องโหว่การสร้างไฟล์ดังกล่าวคือ การ include _CREAT และ O_EXCL [6] ในการเรียก open()

โปรแกรม 2.52 File existence

```
int fd =open("/tmp/some_file",
                                        O_WRONLY | O_CREAT |
O_EXCL | O_TRUNC, 0600);
```

โปรแกรม 2.52 แสดงรูปแบบของโค้ด ที่พบบ่อยซึ่งเป็นวิธีการทดสอบสำหรับ file existence ที่จะเปิดไฟล์ด้วย stream เพราะ การทดสอบ file existence ในบรรทัด 8 และ และ เปิดไฟล์ในบรรทัด 10 ทั้งชื่อไฟล์ที่ใช้ใน code นี้มีช่อโหว่ของ TOCTOU อีกทั้งโค้ดยังสามารถใช้ช่องโหว่สร้างการเชื่องแบบ symbolic link ได้ ชื่อ /tmp/some_file ระหว่าง race window ระหว่างการดำเนินการบนบรรทัดที่ 8 และ 10

โปรแกรม 2.53 Code with TOCTOU vulnerability in file open

```
1.#include <iostream>
2.#include <fstream>
3.#include <string>
4.using namespace std;
5.int main(void){
6. ofstream outStrm;
7. ifstream chkStrm;
8. chkStrm.open("/tmp/some_file", ifstream::in);
9. if (!chkStrm.fail())
```

โปรแกรม 2.54 นี่เป็นรูปแบบ C-style stream ที่ถูกนำมาใช้ในภาษา C++ stream หัวใจสำคัญของการแก้ปัญหาที่พบในบรรทัดที่ 10 ที่ stream ถูกเปิดไม่ได้กับชื่อไฟล์แต่มีการอธิบายไฟล์ไว้ การเปลี่ยนแปลงนี้กับการใช้งานของ O_EXCL กับ open() ฟังก์ชัน(บรรทัดที่ 7) ที่ช่วยลดสภาพ TOCTOU ให้เป็นวิธีที่ปลอดภัยในการทดสอบของ file existence เมื่อกำลังเปิด stream

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรม 2. 54 Mitigation for the TOCTOU vulnerability in file open

```
1.#include <stdio.h>
2.#include <fcntl.h>
3.#include <errno.h>
4.int main(void){
5. int fd;
6. FILE *fp;
7. if ((fd =open("/tmp/some_file",O_EXCL|O_CREAT
|O_TRUNC|O_RDWR, 0600))==-1)
{
8. err(1, "/tmp/some_file");
9. }
10. fp =fdopen(fd, "w");
```

เมื่อการสร้าง temporary file ชื่อของแฟ้มมักจะไม่สำคัญตราบใดที่มันไม่ขัดแย้งกับไฟล์ที่มีอยู่ ผู้โจมตีสามรถที่ใช้ช่องโหว่ของสร้างไฟล์ /tmp ถ้ามันเป็นไปได้ที่จะเดาชื่อไฟล์ก่อนที่จะมีกระบวนการสร้างไฟล์ แม้การใช้งานของ random number generators ในการตั้งชื่อไฟล์อาจไม่ปลอดภัย หากผู้โจมตีสามารถค้นพบขั้นตอนวิธีการได้ วิธีการแก้ปัญหาที่มีความปลอดภัยของ generating random file names ใน UNIX โดยการเรียกใช้ mkstemp() ฟังก์ชันดังนี้

โปรแกรม 2. 55 Use mkstemp()

```
char template[]="/tmp/fileXXXXXX";

      if ((fd =mkstemp(template))==-1){
       err(1, "random file");
                  }
```

การเรียกใช้ mkstemp() จะแทนที่หก Xs ในสตริงแม่แบบที่มี หกตัวอักษรในการสุ่มเลือก ซึ่งการเรียก mkstemp() จะประกันมีความปลอดภัยระดับหนึ่งแต่ก็ยังต้องมีการตรวจสอบข้อผิดพลาดของ mkstemp() อัลกอริทึมสำหรับการเลือกชื่อไฟล์อยู่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.11 Mitigation Strategies

2.11.1 Closing the Race Window

ช่องโหว่ของ race condition จะอยู่ระหว่าง race window ดังนั้นการบรรเทาผลกระทบที่เห็นได้ชัดที่สุดคือการกำจัด race window ที่เป็นไปได้ ในส่วนนี้จะแสดงให้เห็นเทคนิคหลายจุดเพื่อกำจัด race window

Checking File Properties Securely. เป็นกล่าวถึงเงื่อนไขของ TOCTOU ทั่วไปที่เป็นผลมาจากความต้องการที่จะตรวจสอบคุณสมบัติของไฟล์ ใน Linux สำหรับข้อผิดพรากทั่วไป stat () ตามด้วยการใช้ open() และ ตามด้วย fstat() ซึ่งเป็นผลกระทบของ Window ที่คล้ายกันคือการใช้ GetFileInformationByHandle() มากกว่า FindFirtFile() หรือ FindFirtFileEx()
Istat() ฟังก์ชัน poses เป็นปัญหาที่ยาก ฟังก์ชันนี้เป็นฟังก์ชันพิเศษเพราะมันเป็นเพียงฟังก์ชัน stat Linux ที่เป็น symbolic link (เมื่อ) ที่มากกว่าเป้าหมาย แต่ Istat () จะต้องนำไปใช้กับชื่อไฟล์ที่มีการอธิบายไฟล์ เป็นผลให้ Istat() จะต้องเป็นการใช้ประโยชน์ในการดูแลช่องโหว่ symlink โดยทั่วไปของ Istat () จะแสดงในโปรแกรม 2. 56 โค้ดนี่ stat เป็นแบบ some_file แล้วเปิดไฟล์นี้ถ้ามันไม่เป็น symbolic link แต่น่าเสียดายที่ Istat() ในบรรทัดที่ 3 ตามด้วย open() บนบรรทัดที่ 7 จาก TOCTOU เพราะ symbolic link จะได้รับการแนะนำให้รู้จักระหว่างบรรทัดเหล่านี้

โปรแกรม 2. 56 Checking for symbolic link with TOCTOU condition

```
1.struct stat lstat_info;
 2.int fd;
 3.if (lstat("some_file", &lstat_info)==-1){
 4. err(1, "lstat");
 5.}
 6.if (!S_ISLNK(lstat_info.st_mode)){
 7. if ((fd =open("some_file", O_EXCL | O_RDWR, 0600))==-1)
 8.  err(2, "some_file");
 9.}
10.//process the file
 }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กลยุทธ์การบรรเทาผลกระทบของการเรียก Istat() เพื่อความปลอดภัยเป็นขั้นตอนและวิธีการอยู่ 4วิธีด้วยกัน

1) Istat() ชื่อไฟล์

2) open() ไฟล์

3) fstat() อธิบายไฟล์จากขั้นตอนที่ 2

4) เปรียบเทียบสถานะจากขั้นตอนที่ 1 และ 3 เพื่อให้มั่นใจว่าไฟล์ที่มีเหมือนกัน

โปรแกรม 2.57 แสดงให้เห็นถึงวิธีการใช้กลยุทธ์นี้ในการลดสภาพของ TOCTOU ที่อยู่ในโปรแกรม 2.56 ซึ่งโค้ดนี้มีการปิดเงื่อนไข race window โดยมั่นใจว่าไฟล์จะผ่านไปที่ Istat () ในบรรทัดที่ 3 จะเป็นไฟล์เดียวกันกับแฟ้มที่ถูกเปิด นี้ซึ่งมีการประกันได้เนื่องจาก fstat() ถูกนำมาใช้ในการยื่นอธิบายให้ไม่มีชื่อไฟล์ดังนั้นไฟล์ที่ผ่านไป fstat() จะต้องเหมือนกันไปยังแฟ้มที่ถูกเปิดและ Istat() จะไม่ปฏิบัติตาม symbolic link แต่ fstat() จะทำโดยไม่มีการเปรียบเทียบโหมด (st_mode) ในบรรทัดที่ 12 จะเพียงพอที่จะตรวจสอบ symbolic link และ เปรียบเทียบกับ inode(st_ino) เพื่อให้แน่ใจว่าไฟล์ที่มีผ่านการฟังก์ชัน istat() แล้วและ มีไอโหนดเหมือนกับไฟล์ที่ส่งผ่านไปในฟังก์ชัน fstat()

โปรแกรม 2.57 Checking for symbolic link without TOCTOU condition

```
1.struct stat lstat_info, fstat_info;
 2.int fd;
 3.if (lstat("some_file", &lstat_info)==-1){
 4. err(1, "lstat");
 5.}
 6.if ((fd =open("some_file", O_EXCL | O_RDWR,
                        0600))==-1){
 7. err(2, "some_file");
 8.}
 9.if (fstat(fd, &fstat_info)==-1){
10. err(3, "fstat");
11.}
12.if (lstat_info.st_mode ==fstat_info.st_mode
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3

ทฤษฎีการออกแบบและทฤษฎีการพัฒนาสื่อการสอน

3.1 การออกแบบระบบการสอน

การออกแบบระบบการสอน (ISD : Instructional System Design หรือ ID : Instructional Design) หมาย ถึง การจัดระบบการสอนอย่างมีระบบ โดยอาศัยความรู้เกี่ยวกับกระบวนการเรียนรู้ ซึ่งรวบรวมองค์ประกอบและปัจจัย ต่าง ๆ เพื่อนำไปสู่กระบวนการตัดสินใจออกแบบระบบ แล้วจึงทำการทดลองและปรับปรุงแก้ไขจนใช้ได้ผล เป็นการนำไปสู่ความสำเร็จของการเรียนรู้ตามวัตถุประสงค์ที่กำหนดไว้ กระบวนการออกแบบระบบการสอน จะประกอบไปด้วยหลักพื้นฐาน 4 ส่วน ดังต่อไปนี้

1) วัตถุประสงค์ เป็นส่วนที่กำหนดวัตถุประสงค์การเรียนรู้ของผู้เรียน
2) ผู้เรียน โดยพิจารณาคุณสมบัติของผู้เรียน เพื่อการออกแบบระบบการสอนให้เหมาะสม
3) วิธีการและกิจกรรม กำหนดวิธีการและกำหนดกิจกรรมในกระบวนการเรียนรู้ เพื่อให้ผู้เรียนเกิดการเรียนรู้ตามวัตถุประสงค์อย่างมีประสิทธิภาพ
4) การวัดและประเมินผล เป็นการกำหนดวิธีการวัดและประเมินผลการเรียนรู้ของผู้เรียนให้สอดคล้องตามวัตถุประสงค์

สำหรับการนิยามของคำว่าการออกแบบระบบการสอน ได้มีการนิยามไว้เป็นประเด็นดังนี้

Instructional System Design is a Process หมายถึง การออกแบบระบบการสอนเป็นกระบวน การที่มีขั้นตอน โดยใช้วิธีการระบบตามหลักการศึกษาและทฤษฎีการเรียนการสอน เพื่อออกแบบบทเรียนให้มีคุณภาพ แต่ละขั้นตอนจึงมีความสัมพันธ์กันทั้งวัสดุการเรียนและกิจกรรมการเรียน ในขั้นตอนสุดท้ายของการออกแบบการเรียนการสอนส่วนพหญ่จะเป็น ขั้นตอนของการวัดและประเมินผล

Instructional System Design is a Discipline หมายถึง การออกแบบระบบการสอนเป็น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ส่วนหนึ่งของความรู้ที่เกี่ยวกับทฤษฎีการเรียนรู้ต่างๆ ซึ่งมีขั้นตอนการดำเนินการอย่างเป็นระบบและถูกต้อง

Instructional System Design is a Science หมายถึง การออกแบบระบบการสอนเป็นวิทยาศาสตร์ ประกอบด้วยขั้นตอนการออกแบบ การพัฒนา การทดลองใช้ การประเมินผล และการบำรุงรักษา ภายใต้สถานการณ์ที่กำหนดไว้ โดยเป็นกระบวนการทางวิทยาศาสตร์ที่เป็นเหตุเป็นผลซึ่งกันและกัน

Instructional System Design is a Reality หมายถึง การออกแบบระบบการสอนเป็นกระบวนการของความจริงที่สามารถพิสูจน์ได้ เนื่องจากอาศัยหลักการของการใช้เหตุและผลบนพื้นฐานของความจริง โดยยึดหลักการศึกษา

3.2 แนวทางการพัฒนาสื่อการสอน

3.2.1 Ebook (Electronic Book)

ที่ผู้อ่านสามารถอ่านผ่านทางอินเทอร์เน็ต หรือ อุปกรณ์อิเล็กทรอนิกส์พกพาอื่นๆได้ สำหรับหนังสือ หรือเอกสารกอิเล็กทรอนิกส์นี้ จะมีความหมายรวมถึงเนื้อหาที่ถูกดัดแปลง อยู่ในรูปแบบที่สามารถแสดงผลออกมาได้โดยเครื่องมืออิเล็กทรอนิกส์ แต่ก็ให้มีลักษณะการนำเสนอที่สอดคล้องและคล้ายคลึงกับการอ่านหนังสือทั่วๆไปในชีวิตประจำวัน แต่จะมีลักษณะพิเศษ คือ สะดวกและรวดเร็วในการค้นหา และผู้อ่านสามารถอ่านพร้อมๆ กันได้โดยไม่ต้องรอให้อีกฝ่ายส่งคืนห้องสมุด เช่นเดียวกับหนังสือในห้องสมุดทั่วๆ ไป

ประเภทของหนังสืออิเล็กทรอนิกส์

หนังสืออิเล็กทรอนิกส์ แบ่งออกเป็น 10 ประเภท ดังนี้คือ

1) หนังสืออิเล็กทรอนิกส์ หรือแบบตำรา (Textbooks) หนังสือ อิเล็กทรอนิกส์ รูปหนังสือ ปกติที่พบเห็นทั่วไป หลักหนังสืออิเล็กทรอนิกส์ชนิดนี้สามารถกล่าวได้ว่าเป็นการแปลงหนังสือ จาก สภาพสิ่งพิมพ์ปกติเป็นสัญญาณดิจิตอล เพิ่มศักยภาพเดิมการนำเสนอ การปฏิสัมพันธ์ระหว่างหนังสืออีเล็กทรอนิกส์ ด้วยศักยภาพของคอมพิวเตอร์ขั้นพื้นฐาน เช่น การเปิดผู้อ่าน หน้าหนังสือ การสืบค้น การคัดเลือก เป็นต้น
2) หนังสืออีเล็กทรอนิกส์ แบบหนังสือเสียงอ่าน มีเสียงคำอ่าน เมื่อเปิดหนังสือจะมีเสียงอ่านหนังสืออิเล็กทรอนิกส์ประเภทเหมาะสำหรับหนังสือเด็กเริ่มเรียน หรือหนังสือฝึกออกเสียง หรือฝึกพูด (Talking Book1) เป็นต้น หนังสืออิเล็กทรอนิกส์ชนิดนี้เป็นการเน้นคุณลักษณะด้านการนำเสนอเนื้อหาที่ เป็นตัวอักษรและเสียงเป็นคุณลักษณะหลัก นิยมใช้กับกลุ่มผู้อ่านที่มีระดับลักษณะทางภาษาโดยเฉพาะด้านการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ฟังหรือการ อ่านค่อนข่างต่ำ เหมาะสำหรับการเริ่มต้นเรียนภาษาของเด็ก ๆ หรือผู้ที่กำลังฝึกภาษาที่สอง หรือฝึกภาษาใหม่เป็นต้น

3) หนังสืออิเล็กทรอนิกส์แบบหนังสือภาพนิ่ง หรืออัลบั้มภาพ (static Picture Books) เป็นหนังสืออิเล็กทรอนิกส์ ที่มีคุณลักษณะหลักเน้นจัดเก็บข้อมูล และนำเสนอข้อมูลในรูปแบบภาพนิ่ง (static picture) หรืออัลบั้มภาพเป็นหลัก เสริมด้วยการนำศักยภาพของคอมพิวเตอร์มาใช้ในการนำเสนอ เช่น การเลือกภาพที่ต้องการ การขยายหรือย่อขนาดของภาพของคอมพิวเตอร์มาใช้ในการนำเสนอ เช่น การเลือกภาพที่ต้องการ การขยายหรือย่อขนาดของภาพหรือตัวอักษร การสำเนาหรือการถ่ายโอนภาพ การแต่งเติมภาพ การเลือกเฉพาะส่วนของภาพ (cropping) หรือเพิ่มข้อมูล เชื่อมโยงภายใน (Linking information) เช่น เชื่อมข้อมูลอธิบายเพิ่มเติม เชื่อมข้อมูลเสียงประกอบ เป็นต้น

4) หนังสืออิเล็กทรอนิกส์ แบบ หนังสือภาพเคลื่อนไหว (Moving Picture Books) เป็นหนังสืออิเล็กทรอนิกส์ที่เน้น การนำเสนอข้อมูลในรูปแบบภาพวีดิทัศน์ (Video Clips) หรือภาพยนตร์สั้น ๆ (Films Clips) ผนวกกับข้อมูลสนเทศที่อยู่ในรูปตัวหนังสือ (Text Information) ผู้อ่านสามารถเลือกชมศึกษาข้อมูลได้ ส่วนใหญ่นิยมนำเสนอข้อมูลเหตุการณ์ประวัติศาสตร์ หรือเหตุการณ์สำคัญ เช่น ภาพเหตุการณ์สงครามโลก ภาพการกล่าวสุนทรพจน์ของบุคคลสำคัญ ๆ ของโลกในโอกาสต่าง ๆ ภาพเหตุการณ์ความสำเร็จหรือสูญเสียของโลกเป็นต้น

5) หนังสืออิเล็กทรอนิกส์แบบหนังสือสื่อประสม (Multimedia) เป็นหนังสืออิเล็กทรอนิกส์ที่เน้นเสนอข้อมูลเนื้อหาสาระ ในลักษณะแบบสื่อผสมระหว่างสื่อภาพ (Visual Media) เป็นทั้งภาพนิ่งและภาพเคลื่อนไหวกับสื่อประเภทเสียง (Audio Media)ในลักษณะต่าง ๆ ผนวกกับศักยภาพของคอมพิวเตอร์อื่นเช่นเดียวกับหนังสืออิเล็กทรอนิกส์อื่น ๆ ที่กล่าวมาแล้ว

6) หนังสืออิเล็กทรอนิกส์แบบหนังสือสื่อหลากหลาย (Polymedia books) เป็น หนังสืออิเล็กทรอนิกส์ที่มีลักษณะเช่นเดียวกับหนังสืออิเล็กทรอนิกส์แบบสื่อ ประสม แต่มีความหลากหลายในคุณลักษณะด้านความเชื่อมโยงระหว่างข้อมูลภายในเล่มที่บันทึกในลักษณะต่าง ๆ เช่น ตัวหนังสือภาพนิ่ง ภาพเคลื่อนไหว เสียงดนตรี และอื่น ๆ เป็นต้น

7) หนังสืออิเล็กทรอนิกส์แบบหนังสือเชื่อมโยง (Hypermedia Book) เป็น หนังสือที่มีคุณลักษณะสามารถเชื่อมโยงเนื้อหาสาระภายในเล่ม (Internal Information Linking) ซึ่งผู้อ่านสามารถคลิกเพื่อเชื่อมไปสู่เนื้อหาสาระที่ออกแบบเชื่อมโยงกันภาย ใน ภาร เชื่อมโยงเช่นนี้มีคุณลักษณะเช่นเดียวกับบทเรียนโปรแกรมแบบแตกกิ่ง (Branching

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Programmed Instruction)นอกจากนี้ยังสามารถเชื่อมโยงกับแหล่งเอกสารภายนอก (External or Information Sources) เมื่อเชื่อมต่อระบบอินเตอร์เน็ต

8) หนังสืออิเล็กทรอนิกส์แบบหนังสืออัจฉริยะ (Intelligent Electronic Books) เป็นหนังสือประสม แต่มีการใช้โปรแกรมชั้นสูงที่สามารถมีปฏิกิริยา หรือ ปฏิสัมพันธ์กับผู้อ่านเสมือนหนังสือมีสติปัญญา (อัจฉริยะ) ในการไตร่ตรอง หรือคาดคะเนในการโต้ตอบ หรือปฏิกิริยากับผู้อ่าน

9) หนังสืออิเล็กทรอนิกส์ แบบสื่อหนังสือทางไกล (Telemedia Electronic Books) หนังสืออิเล็กทรอนิกส์ประเภทนี้มีคุณลักษณะหลักต่าง ๆ คล้ายกับ Hypermedia Electronic Booksแต่เน้นการเชื่อมโยงกับแหล่งข้อมูลภายนอกผ่านระบบเครือข่าย (Online Information Sourcess) ทั้งที่เป็นเครือข่ายเปิด และเครือข่ายเฉพาะสมาชิกของเครือข่าย

10) หนังสืออิเล็กทรอนิกส์แบบหนังสือไซเบอร์เสปซ (Cyberspace books) หนังสืออิเล็กทรอนิกส์ประเภทนี้มีลักษณะเหมือนกับหนังสืออิเล็กทรอนิกส์หลาย ๆ แบบที่กล่าวมาแล้วผสมกัน สามารถเชื่อมโยงแหล่งข้อมูลทั้งจากแหล่งภายในและภายนอก สามารถนำเสนอข้อมูลใน ระบบสื่อที่หลากหลาย สามารถปฏิสัมพันธ์กับผู้อ่านได้หลากหลายนั้นเอง

3.2.2 WBI (Web-based Instruction)

WBI หรือ Web Base Instruction เป็นการจัดกิจกรรมการสอนในรูปแบบของ Web Knowledge Based โดยใช้เทคโนโลยีทางของ Webpage เป็นศูนย์กลางในการนำเสนอเนื้อหา ประเภทของการเรียนการสอนผ่านเว็บแบ่งตามลักษณะของการสื่อสารได้ดังนี้

รูปแบบการเผยแพร่ รูปแบบนี้สามารถแบ่งได้ออกเป็น 3 ชนิด คือ

1) รูปแบบห้องสมุด (Library Model) เป็น รูปแบบที่ใช้ประโยชน์จากความสามารถในการเข้าไปยังแหล่งทรัพยากร อิเล็กทรอนิกส์ที่มีอยู่หลากหลาย โดยวิธีการจัดหาเนื้อหาให้ผู้เรียนผ่านการเชื่อมโยงไปยังแหล่งเสริมต่างๆ เช่นสารานุกรม วารสาร หรือหนังสือออนไลน์ทั้งหลาย ซึ่งถือได้ว่า เป็นการนำเอาลักษณะทางกายภาพของห้องสมุดที่มีทรัพยากรจำนวนมหาศาลมาประยุกต์ ใช้ ส่วน ประกอบของรูปแบบนี้ได้แก่ สารานุกรมออนไลน์ วารสารออนไลน์ หนังสือออนไลน์ สารบัญการอ่านออนไลน์ (Online Reading List) เว็บห้องสมุด เว็บงานวิจัย รวมทั้งการรวบรวมรายชื่อเว็บที่สัมพันธ์กับวิชาต่างๆ

2) รูปแบบหนังสือเรียน (Textbook Model) การเรียนการสอนผ่านเว็บรูปแบบนี้เป็นการจัดเนื้อหาของหลักสูตรในลักษณะออนไลน์ให้แก่ผู้เรียน เช่น คำบรรยาย สไลด์ นิยาม คำศัพท์และส่วนเสริมผู้สอนสามารถเตรียมเนื้อหาออนไลน์ที่ใช้เหมือนกับที่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ใช้ ในการเรียนในชั้นเรียนปกติและสามารถทำสำเนาเอกสารให้กับผู้เรียนได้ รูปแบบนี้ต่างจากรูปแบบห้องสมุดคือรูปแบบนี้จะเตรียมเนื้อหาสำหรับการเรียนการสอนโดยเฉพาะ ขณะที่รูปแบบห้องสมุดช่วยให้ผู้เรียนเข้าถึงเนื้อหาที่ต้องการจากการเชื่อม โยงที่ได้เตรียมเอาไว้ ส่วนประกอบของรูปแบบหนังสือเรียนนี้ประกอบด้วยบันทึกของหลักสูตร บันทึกคำบรรยาย ข้อแนะนำของห้องเรียน สไลด์ที่นำเสนอ วิดีโอและภาพที่ใช้ในชั้นเรียน เอกสารอื่นที่มีความสัมพันธ์กับชั้นเรียน เช่น ประมวลรายวิชา รายชื่อในชั้น กฎเกณฑ์ข้อตกลงต่าง ๆ ตารางการสอบและตัวอย่างการสอบครั้งที่แล้ว ความคาดหวังของชั้นเรียน งานที่มอบหมาย เป็นต้น

3) รูปแบบการสื่อสาร (Communication Model)การเรียนการสอนผ่านเว็บรูปแบบนี้เป็นรูปแบบที่อาศัยคอมพิวเตอร์มาเป็นสื่อเพื่อการสื่อสาร (Computer – Mediated Communications Model) ผู้ เรียนสามารถที่จะสื่อสารกับผู้เรียนคนอื่นๆ ผู้สอนหรือกับผู้เชี่ยวชาญได้ โดยรูปแบบการสื่อสารที่หลากหลายในอินเทอร์เน็ต ซึ่งได้แก่ จดหมาย อิเล็กทรอนิกส์ กลุ่มอภิปรายการสนทนาและการอภิปรายและการประชุมผ่านคอมพิวเตอร์ เหมาะ สำหรับการเรียนการสอนที่ต้องการส่งเสริมการสื่อสารและปฏิสัมพันธ์ระหว่างผู้ ที่มีส่วนร่วมในการเรียนการสอน

4) รูปแบบผสม (Hybrid Model)รูปแบบการเรียนการสอนผ่านเว็บรูปแบบนี้เป็นการนำเอารูปแบบ 2 ชนิด คือ รูปแบบการเผยแพร่กับรูปแบบการสื่อสารมารวมเข้าไว้ด้วยกัน เช่น เว็บไซต์ที่รวมเอารูปแบบห้องสมุดกับรูปแบบหนังสือเรียนไว้ด้วยกัน เว็บไซต์ที่รวบรวมเอาบันทึกของหลักสูตรรวมทั้งคำบรรยายไว้กับกลุ่มอภิปรายหรือเว็บไซต์ที่รวมเอารายการแหล่งเสริมความรู้ต่างๆ และความสามารถของจดหมายอิเล็กทรอนิกส์ไว้ด้วยกัน เป็นต้นรูปแบบนี้มีประโยชน์เป็นอย่างมากกับผู้เรียนเพราะผู้เรียนจะได้ใช้ ประโยชน์ของทรัพยากรที่มีในอินเทอร์เน็ตในลักษณะที่หลากหลาย

5) รูปแบบห้องเรียนเสมือน (Virtual classroom model)รูปแบบห้องเรียนเสมือนเป็นการนำเอาลักษณะเด่นหลายๆ ประการของแต่ละรูปแบบที่กล่าวมาแล้วข้างต้นมาใช้ ฮิลทซ์ (Hiltz, 1993) ได้ นิยามว่าห้องเรียนเสมือนเป็นสภาพแวดล้อมการเรียนการสอนที่นำแหล่งทรัพยากรออ นไลน์มาใช้ในลักษณะการเรียนการสอนแบบร่วมมือ โดยการร่วมมือระหว่างนักเรียนด้วยกัน นักเรียนกับผู้สอน ชั้นเรียนกับสถาบันการศึกษาอื่น และกับชุมชนที่ไม่เป็นเชิงวิชาการ (Khan, 1997) ส่วนเทอรอฟฟ์ (Turoff, 1995)กล่าว ถึงห้องเรียนเสมือนว่า เป็นสภาพแวดล้อมการเรียน การสอนที่ตั้งขึ้นภายใต้ระบบการสื่อสารผ่านคอมพิวเตอร์ในลักษณะของการเรียน แบบร่วมมือ ซึ่งเป็นกระบวนการที่เน้นความสำคัญของกลุ่มที่จะร่วมมือทำกิจกรรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ร่วมกัน นักเรียนและผู้สอนจะได้รับความรู้ใหม่ๆ จากกิจกรรมการสนทนาแลกเปลี่ยนความคิดเห็นและข้อมูล ลักษณะเด่นของการเรียนการสอนรูปแบบนี้ก็คือความสามารถในการลอกเลียนลักษณะ ของห้องเรียนปกติมาใช้ในการออกแบบการเรียนการสอนผ่านเครือข่ายอินเทอร์เน็ต โดยอาศัยความสามารถต่างๆ ของอินเทอร์เน็ต โดยมีส่วนประกอบคือ ประมวลรายวิชา เนื้อหาในหลักสูตร รายชื่อแหล่งเนื้อหาเสริม กิจกรรมระหว่าง ผู้เรียนผู้สอน คำแนะนำและการให้ผลป้อนกลับ การนำเสนอในลักษณะมัลติมีเดีย การเรียนแบบร่วมมือ รวมทั้งการสื่อสารระหว่างกัน รูปแบบนี้จะช่วยให้ผู้เรียนได้รับประโยชน์จากการเรียน โดยไม่มีข้อจำกัดในเรื่องของเวลาและสถานที่

3.2.3 CAI (Computer Assisted Instruction)

คอมพิวเตอร์ ช่วยสอน (Computer Assisted Instruction : CAI) เป็นกระบวนการเรียนการสอน โดยใช้สื่อคอมพิวเตอร์ CAI ย่อมาจากคำว่า COMPUTER- ASSISTED หรือ AIDEDINSTRUCTION คอมพิวเตอร์ช่วยสอน (CAI) หมายถึง สื่อการเรียนการสอนทางคอมพิวเตอร์รูปแบบหนึ่ง ซึ่งใช้ความสามารถของคอมพิวเตอร์ในการนำเสนอสื่อประสมอันได้แก่ ข้อความ ภาพนิ่ง กราฟิก แผนภูมิ กราฟ วิดีทัศน์ ภาพเคลื่อนไหว และเสียง เพื่อถ่ายทอดเนื้อหาบทเรียน หรือองค์ความรู้ในลักษณะที่ ใกล้เคียงกับการสอนจริงในห้องเรียนมากที่สุด

โดย มีเป้าหมายที่สำคัญก็คือ สามารถดึงดูดความสนใจของผู้เรียน และกระตุ้นให้เกิดความต้องการที่ จะเรียนรู้ คอมพิวเตอร์ช่วยสอนเป็นตัวอย่างที่ดีของสื่อการศึกษาในลักษณะตัวต่อตัวในการ นำเสนอเนื้อหาเรื่องราวต่างๆ มีลักษณะเป็นการเรียนโดยตรง และเป็นการเรียน แบบมีปฏิสัมพันธ์ (Interactive) คือสามารถโต้ตอบระหว่างผู้เรียนกับคอมพิวเตอร์ได้ ซึ่งผู้เรียนเกิดการเรียนรู้จากการมีปฏิสัมพันธ์ หรือการโต้ตอบพร้อมทั้งการได้รับผลป้อนกลับ (FEEDBACK) นอกจากนี้ยังเป็นสื่อ ที่สามารถตอบสนองความแตกต่างระหว่างผู้เรียนได้เป็นอย่างดี รวมทั้งสามารถที่จะประเมิน และตรวจสอบความเข้าใจของผู้เรียนได้ตลอดเวลา

ประเภทของบทเรียน CAI

1) สอนเนื้อหารายละเอียด (Tutorials)โปรแกรม ช่วยสอนเนื้อหารายละเอียด หมายถึงโปรแกรมคอมพิวเตอร์ที่ช่วยให้นักเรียน ได้ เรียนรู้ เนื้อหาหรือหลักการใหม่ๆ ด้วยการเสนอเนื้อหาและคำถามคำตอบระหว่างบทเรียน และนักเรียน โปรแกรมจะแสดงเนื้อหาที่จะสอนแล้วตั้งคำถามให้ นักเรียนตอบต่อจากนั้น โปรแกรมจะวิเคราะห์คำตอบแล้วตัดสินว่า จะแสดงเนื้อหาต่อไปหรือให้นักเรียนตอบคำถามใหม่ หรือจะแสดงคำอธิบายเนื้อหาเพิ่มเติม และโปรแกรมช่วยสอนนี้ยังรวมถึงวิธีการแนะนำให้ นักเรียนตัดสินใจแก้ปัญหาอย่างใดอย่างหนึ่ง ด้วยการให้แนวทางแก่นักเรียนเพื่อเลือกคำตอบ ที่ถูกต้อง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2) การฝึกทักษะ (Drill and Practice)หลัง จากที่นักเรียนได้เรียนเนื้อหารายละเอียดแล้ว สิ่งจำเป็นคือการมีโอกาสได้ฝึกทักษะหรือฝึกปฏิบัติซ้ำๆ เพื่อที่จะนำความรู้ที่ได้เรียนแล้วไปใช้ได้อย่างคล่องแคล่วรวดเร็วหรือที่เรียกกันว่าใช้ได้โดย อัตโนมัติก่อน โปรแกรมการฝึกทักษะอาจเน้นการฝึกปฏิบัติเพื่อให้เกิดทักษะเฉพาะอย่าง เช่น ทักษะการบวกเลขทักษะด้านคำศัพท์ ทักษะการอ่านแผนที่ เป็นต้น โปรแกรมประเภทนี้นิยมใช้กันมากในวิชาคณิตศาสตร์ การเรียนภาษา หรือภาษาต่างประเทศ การฝึกทักษะเหล่านี้มักจะใช้คำถามเป็นจำนวนมากซึ่งบาง ครั้งเรียกว่าคลังข้อคำถาม(Item Pool) นอกจากนี้ข้อคำถามที่ ดีควรได้ ผ่านการวิเคราะห์ค่าสถิติ เช่น ระดับความยาก-ง่าย อำนาจจำแนก เป็นต้น โปรแกรมการฝึกทักษะที่ดีควรมีการประเมินข้อบกพร่องของนักเรียนว่าจำเป็นต้อง ฝึกหัด ที่ระดับความรู้ระดับใด และบอกสาเหตุของความบกพร่องในการตอบผิด

3) การจำลองสถานการณ์ (Simulations)โปรแกรมการจำลองสถานการณ์ในการเรียนการสอน เป็นวิธีการเลียนแบบ หรือสร้างสถานการณ์เพื่อทดแทนสภาพจริงในชีวิตประจำวัน สำหรับการเรียนรู้ในชั้นเรียน เพื่อสร้างแรงจูงใจให้นักเรียนเนื่องจากในบางครั้งการฝึกและทดลองจริงอาจมีราคา แพง หรือมีความเสี่ยงอันตรายสูง เช่น การจำลองสถานการณ์การบิน การจำลองการเกิด ปฏิกิริยาของนิวเคลียร์ หรือการจำลองการทำงานของแผงวงจร ไฟฟ้า เป็นต้นซึ่งการจำลองสถานการณ์ทำให้นักเรียนมีส่วนร่วมด้วย เช่นการควบคุมเหตุการณ์การตัดสินใจ การโต้ตอบกับสิ่งที่เกิดขึ้นในสถานการณ์จำลองได้โดยที่ในชีวิตจริง นักเรียนไม่อาจสามารถแสดงปฏิกิริยาเหล่านี้ ได้ อย่างไรก็ตามในสถานการณ์จำลอง ย่อมลดความยุ่ งยากซับซ้อนให้น้อยกว่าเหตุการณ์จริงเช่น ลดรายละเอียด ลดโอกาสที่จะเกิดขึ้น เป็นต้น และในสถานการณ์จำลองนี้นักเรียนต้องแก้ไขปัญหา โดยการเรียนรู้ขั้นตอนกระบวนการด้วยตนเองจนเกิดความเข้าใจในคุณลักษณะต่างๆ ในที่สุด รวมทั้งการเรียนรู้วิธีการควบคุมเหตุการณ์ เหล่านั้น หรือเรียนรู้ว่า จะต้องปฏิ บัติอย่างไรในสถานการณ์ที่แตกต่างกัน

4) เกมการสอน (Instructional games)นอก จากนี้การใช้เกมยังช่วยเพิ่มบรรยากาศในการเรียนรู้ให้ดีขึ้นเนื่องจากมีภาพ แสงสี เสียงและกราฟิกที่มีการเคลื่อนไหวได้ จึงทำให้นักเรี ยนตื่นตัวอยู่เสมอ รูปแบบของ โปรแกรมเกมเพื่อการสอนคล้ายคลึงกับโปรแกรมบทเรียนสถานการณ์ จำลองแต่แตกต่าง กัน โดยการเพิ่มบทบาทของนักเรียนเข้าไปในการใช้โปรแกรมเกมการสอนด้วย

5) การสาธิต (Demonstration)โปรแกรมการสาธิต มีจุดประสงค์ เพื่อสาธิตประกอบการสอน หรือบรรยายเนื้อหาหัวข้อใดหัวข้อหนึ่งเพื่อช่วยผู้เรียนเข้าใจสิ่งที่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

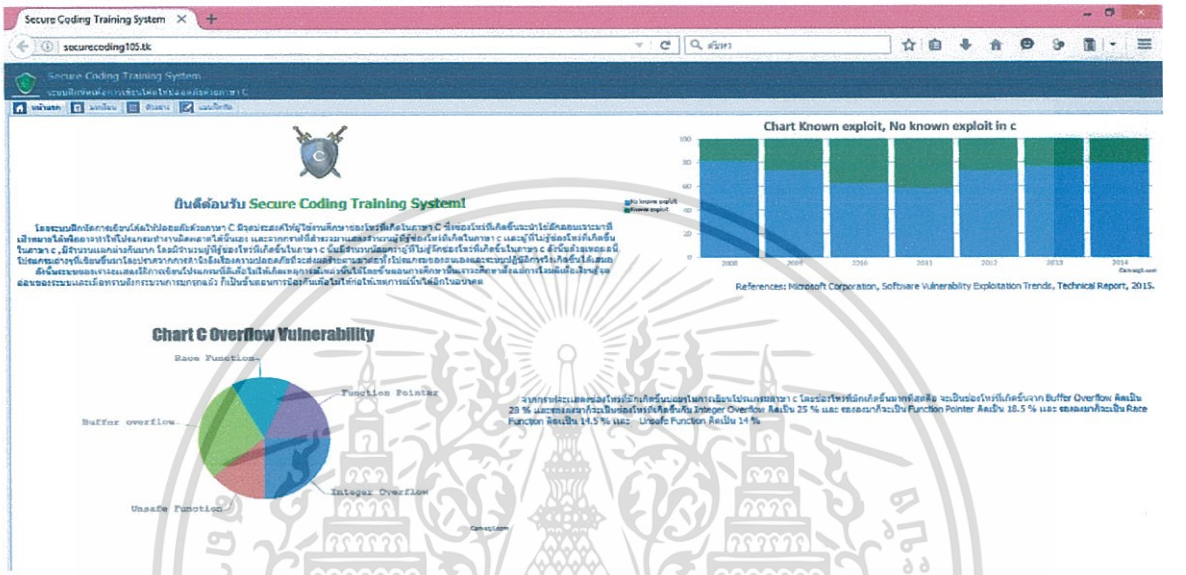
เรียนได้ดียิ่งขึ้น เช่น การเขียนกราฟแสดงรายละเอียด การสาธิตการเกิดสุริยุปราคา หรือสาธิตการ โคจรของดวงดาว การเจริญเติบโตของพืช เป็นต้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4

การทดลองและการประเมินผล

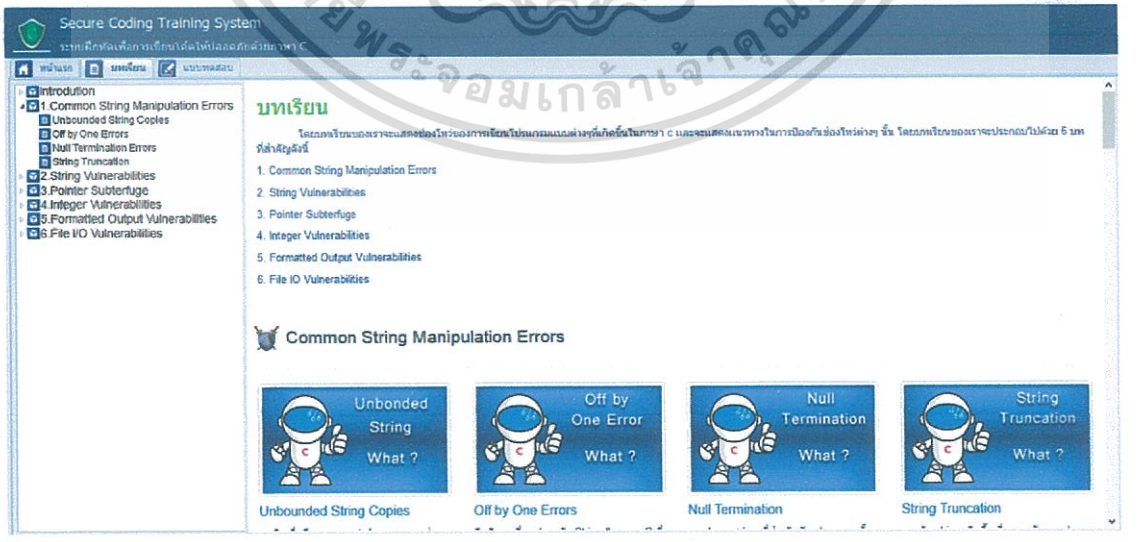
4.1 การทดลองใช้งานส่วนของเว็บไซต์



รูป 4.1 หน้าหลักของเว็บไซต์

4.1.1 ทดลองใช้เว็บในส่วนการเรียนการสอน

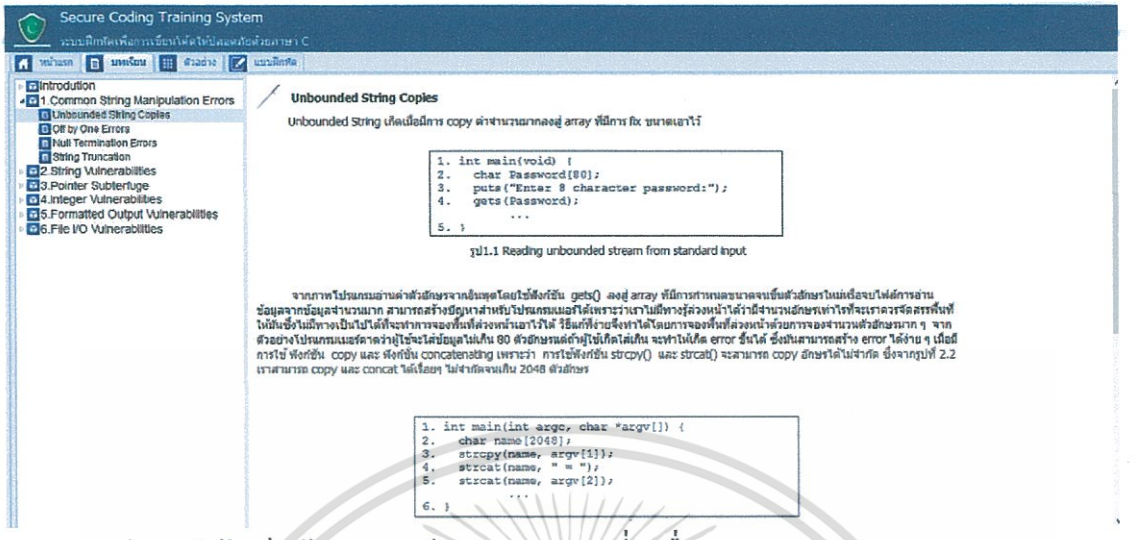
เมื่อผู้ใช้คลิกเข้าสู่บทเรียนจะแสดงหน้าบทเรียนเพื่อให้ใช้เลือกเพื่อเข้าศึกษาบทเรียน



รูป 4.2 หน้าหลักของบทเรียน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

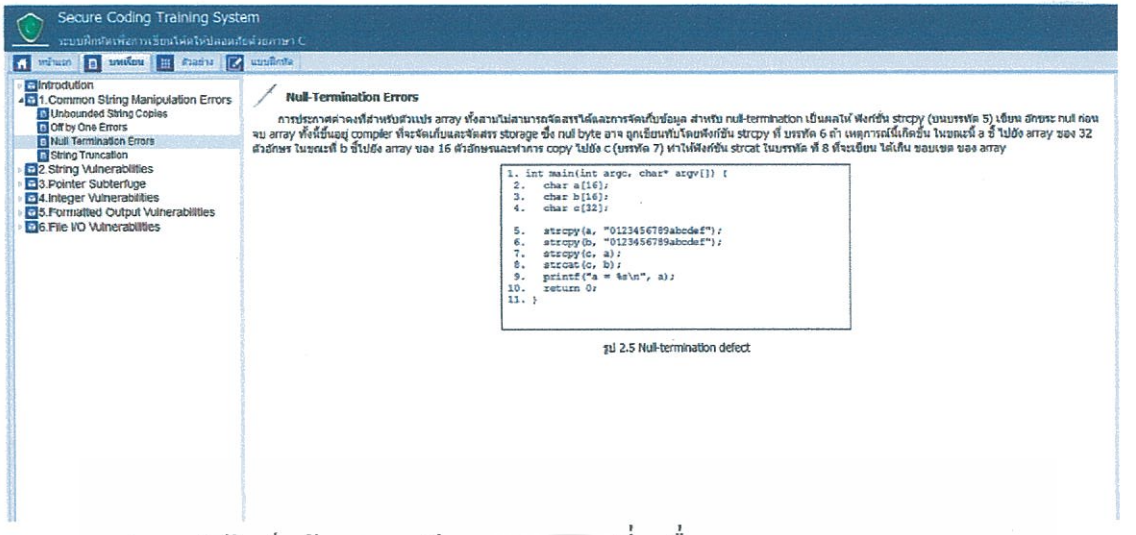
เมื่อผู้ใช้เลือกบทเรียนแล้ว



รูป 4.3 เว็บไซต์หน้าของการเรียนการสอนบทที่ 1 เรื่อง Unbounded String Copies

รูป 4.4 เว็บไซต์หน้าของการเรียนการสอนบทที่ 1 เรื่อง Off-by-One Errors

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

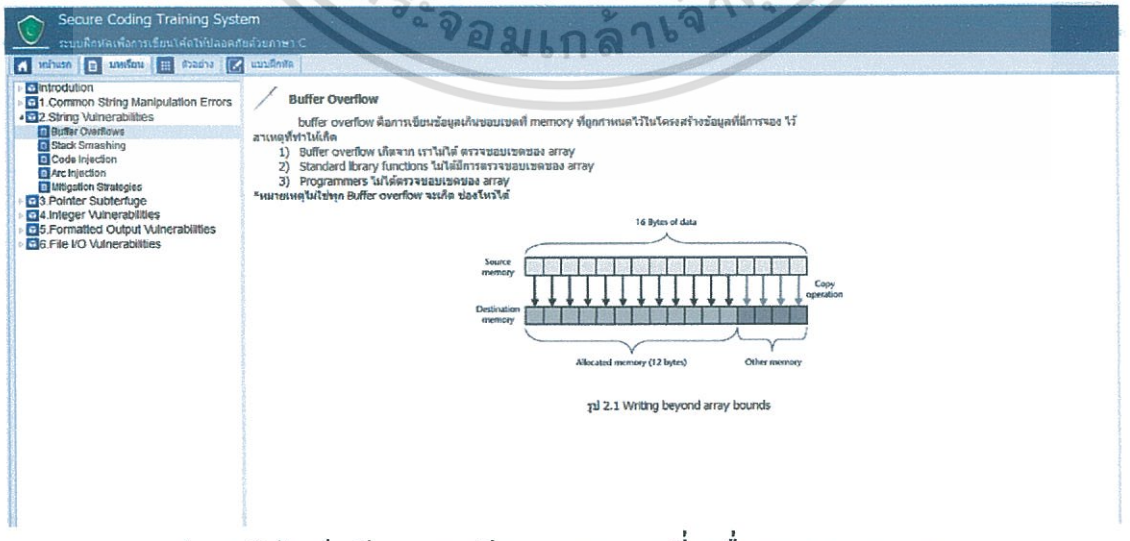


รูป 2.5 Null-termination defect

รูป 4.5 เว็บไซต์หน้าของการเรียนการสอนบทที่ 1 เรื่อง Null-Termination Errors

รูป2.1 Defective string manipulation code

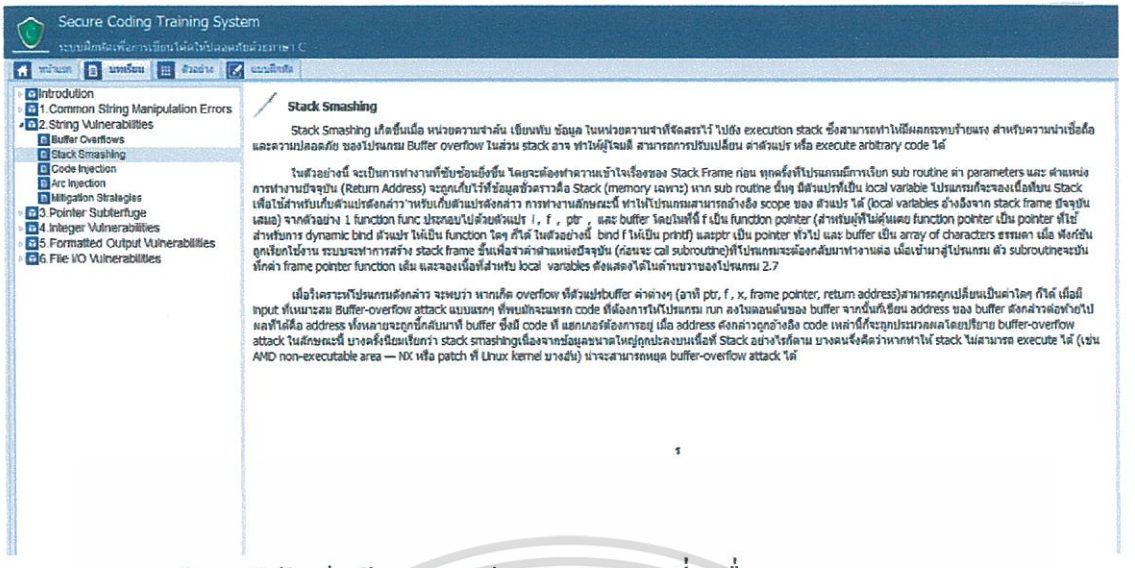
รูป 4.6 เว็บไซต์หน้าของการเรียนการสอนบทที่ 1 เรื่อง String Truncation



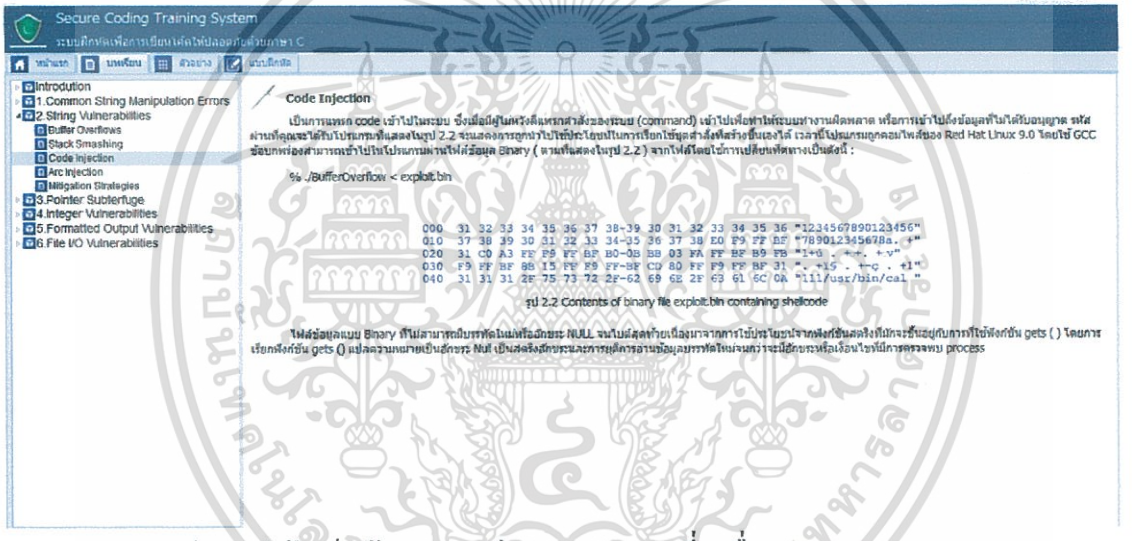
รูป 2.1 Writing beyond array bounds

รูป 4.7 เว็บไซต์หน้าของการเรียนการสอนบทที่ 2 เรื่อง Buffer Overflow

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

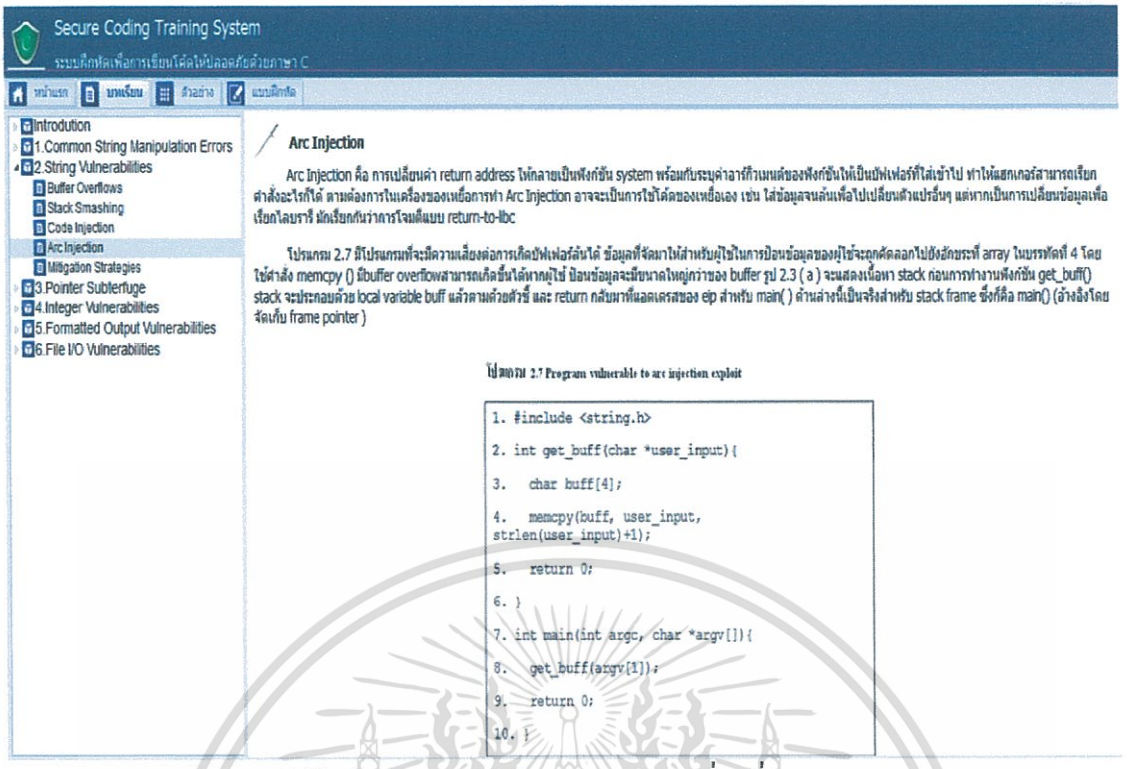


รูป 4.8 เว็บไซต์หน้าของการเรียนการสอนบทที่ 2 เรื่อง Stack Smashing

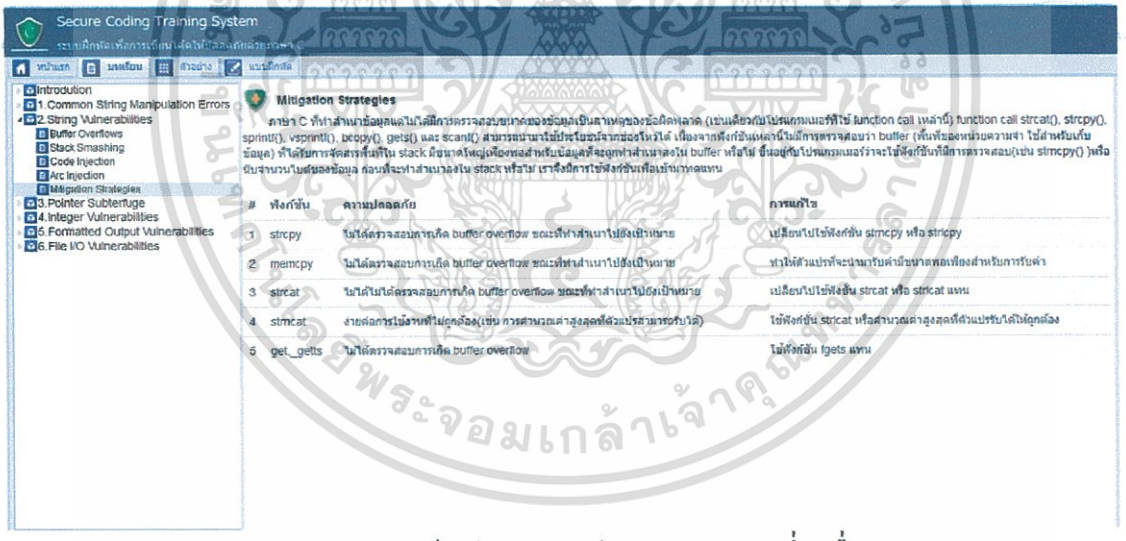


รูป 4.9 เว็บไซต์หน้าของการเรียนการสอนบทที่ 2 เรื่อง Code Injection

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

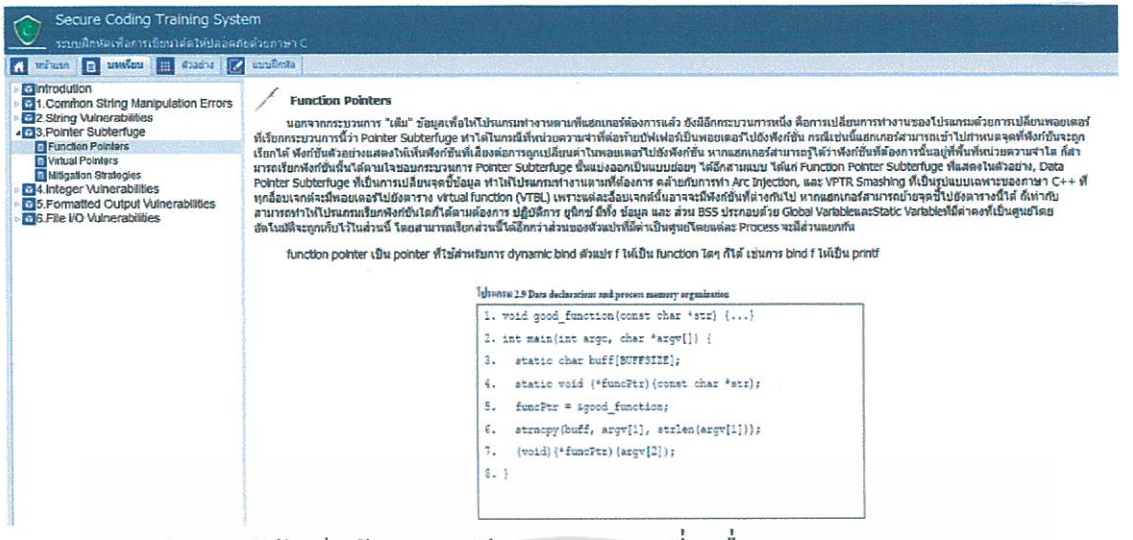


รูป 4.10 เว็บไซต์หน้าของการเรียนการสอนบทที่ 2 เรื่อง Arc Injection

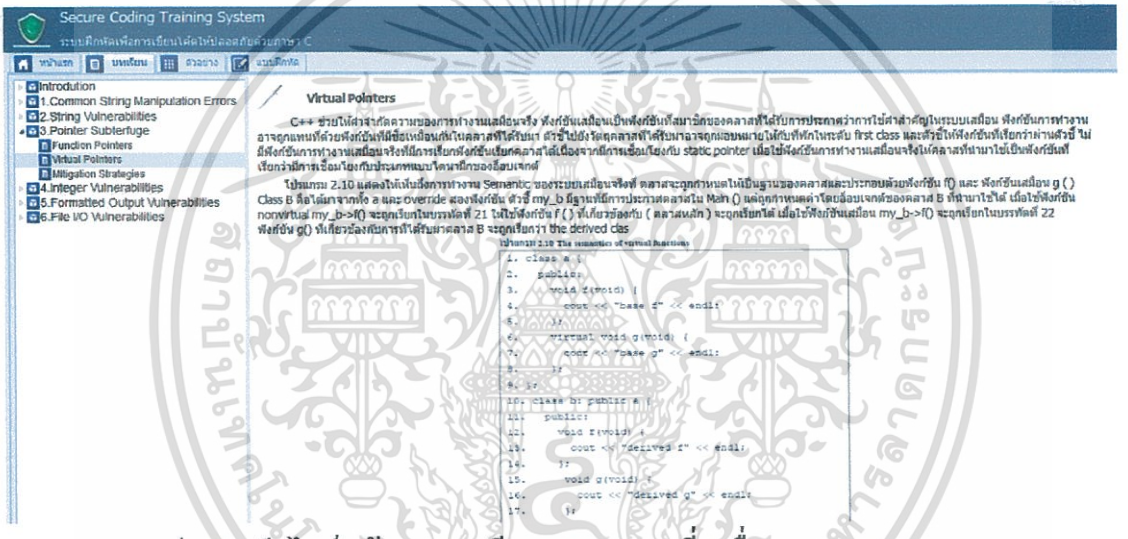


รูป 4.11 เว็บไซต์หน้าของการเรียนการสอนบทที่ 2 เรื่อง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

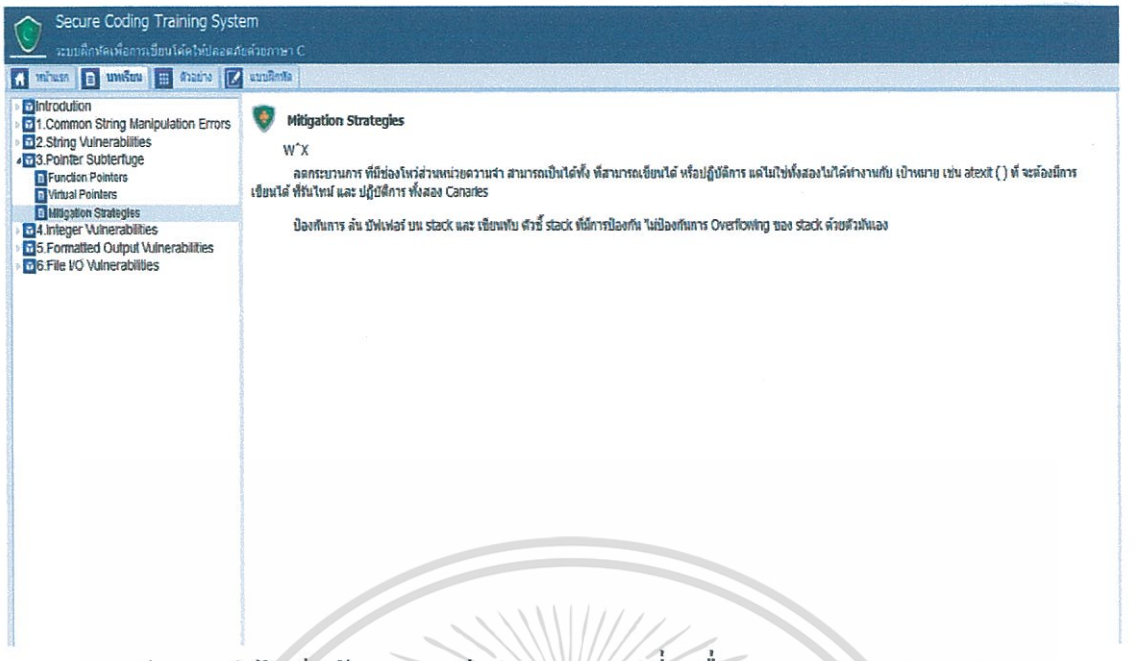


รูป 4.12 เว็บไซต์หน้าของการเรียนการสอนบทที่ 3 เรื่อง Function Pointers

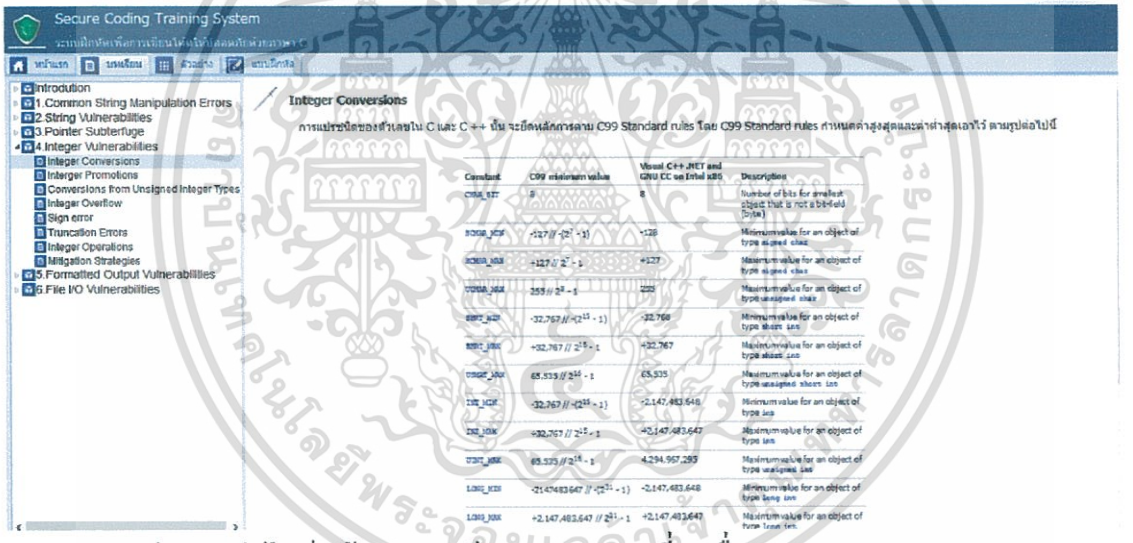


รูป 4.13 เว็บไซต์หน้าของการเรียนการสอนบทที่ 3 เรื่อง Virtual Pointers

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

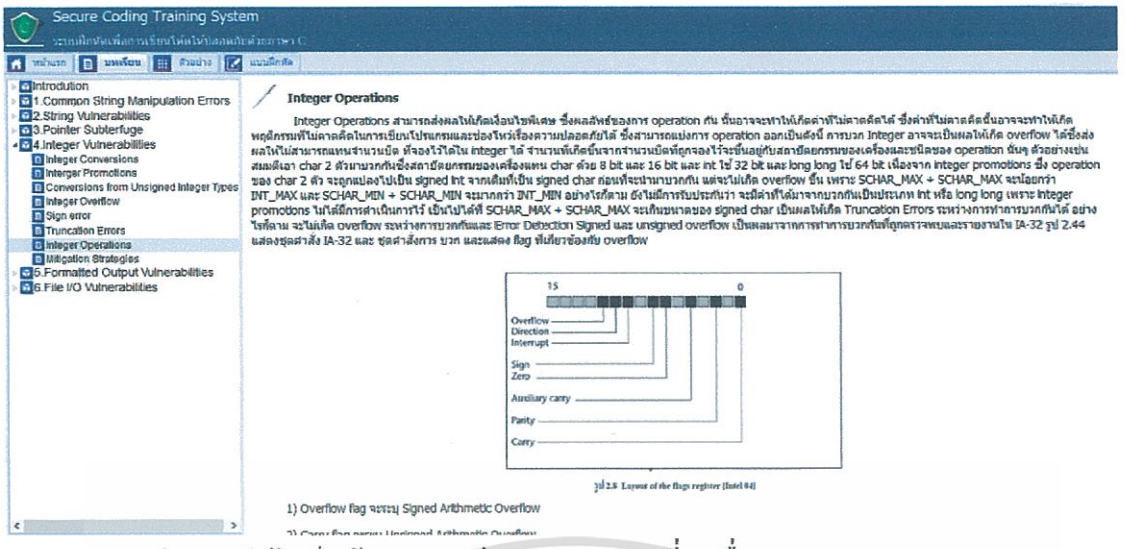


รูป 4.14 เว็บไซต์หน้าของการเรียนการสอนบทที่ 3 เรื่อง Mitigation Strategies

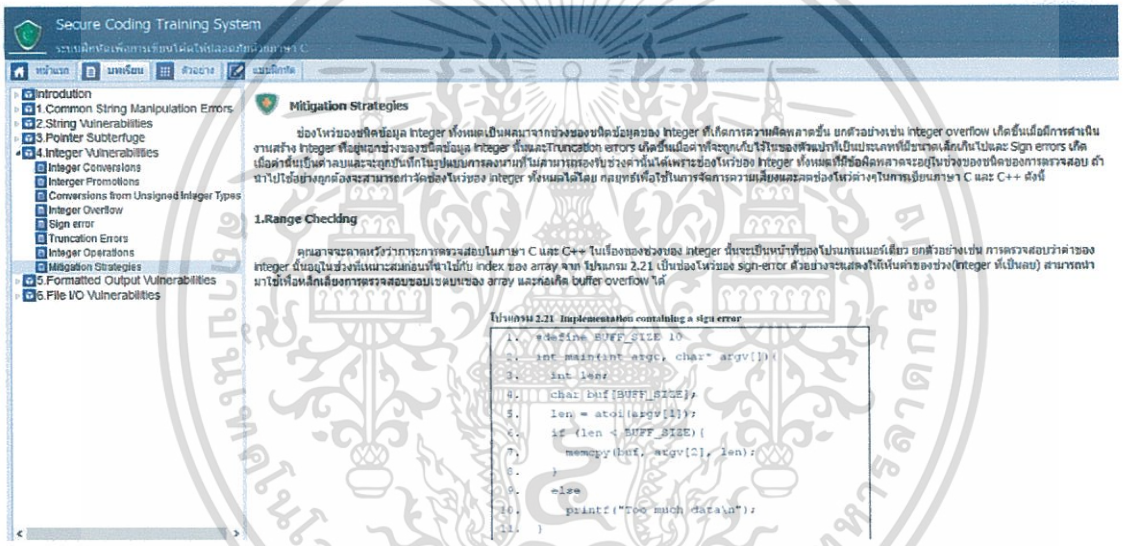


รูป 4.15 เว็บไซต์หน้าของการเรียนการสอนบทที่ 4 เรื่อง Integer Conversion

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

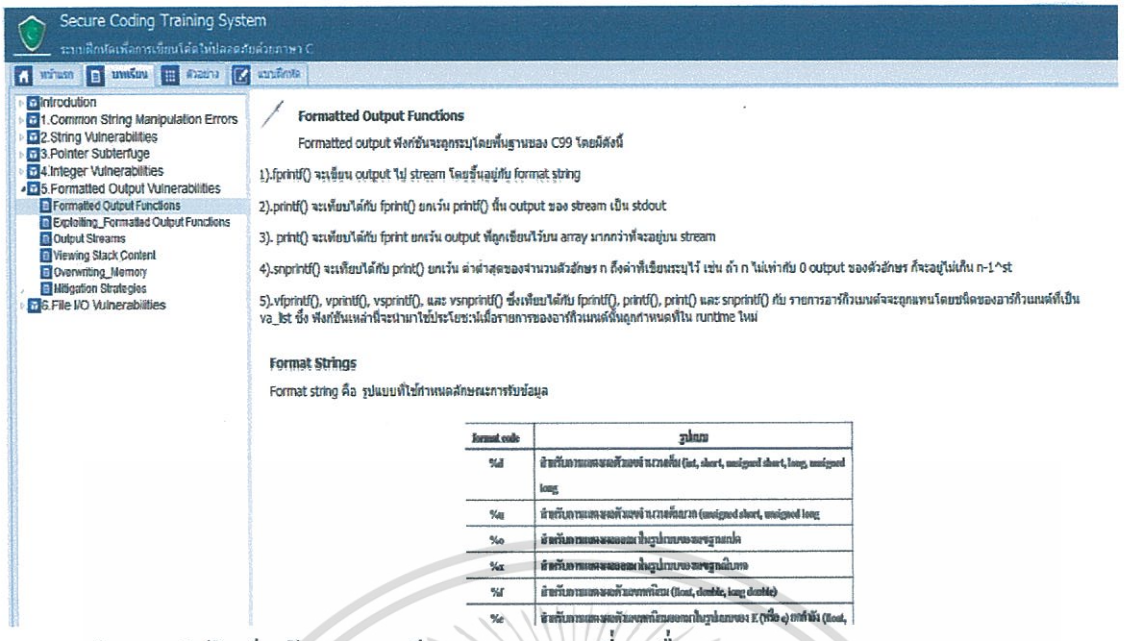


รูป 4.16 เว็บไซต์หน้าของการเรียนการสอนบทที่ 4 เรื่อง Integer Operation

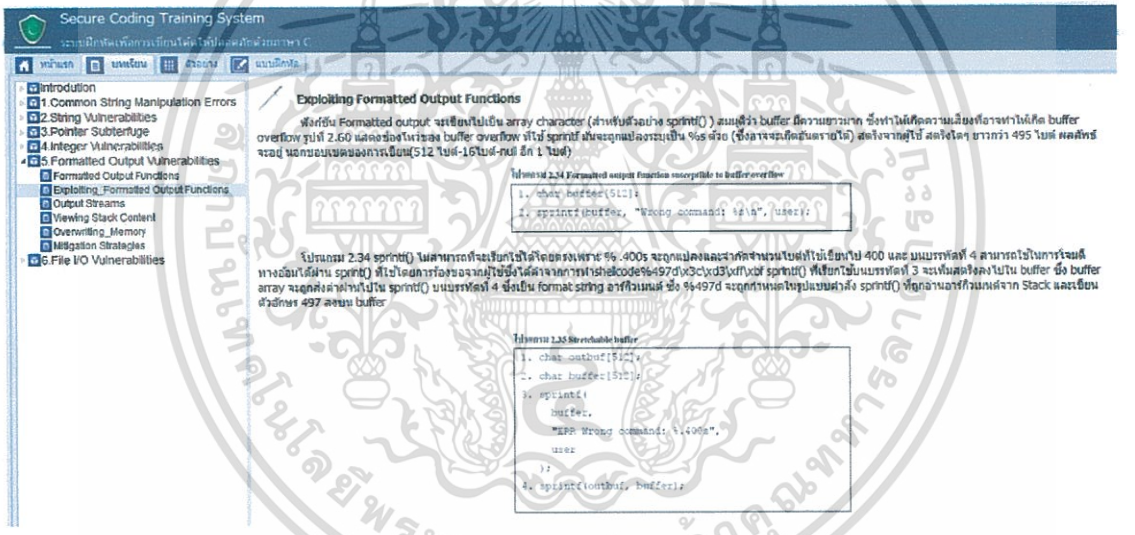


รูป 4.17 เว็บไซต์หน้าของการเรียนการสอนบทที่ 4 เรื่อง Mitigation Strategies

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

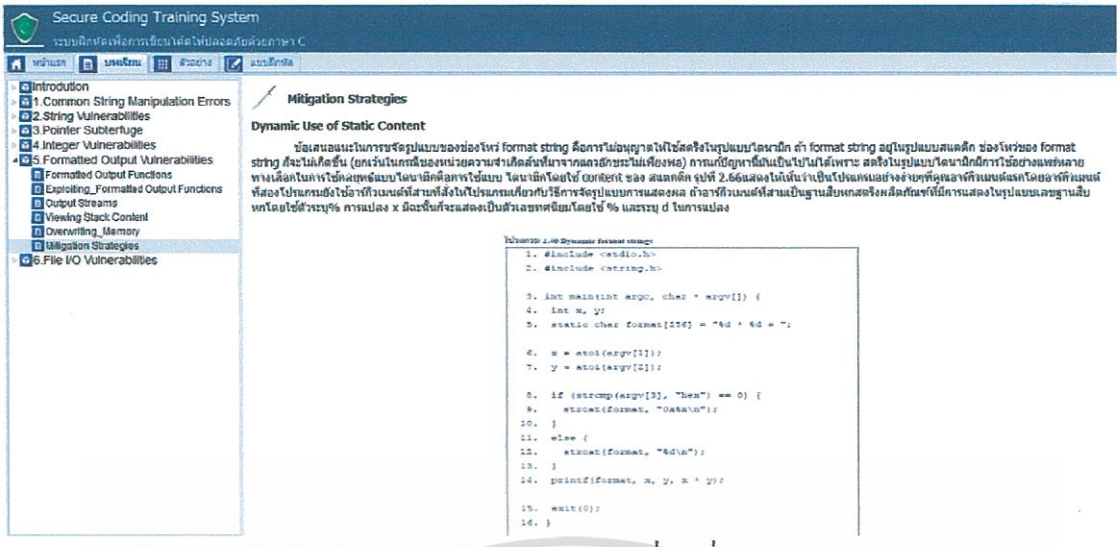


รูป 4.18 เว็บไซต์หน้าของการเรียนการสอนบทที่ 5 เรื่อง Formatted Output Functions



รูป 4.19 เว็บไซต์หน้าของการเรียนการสอนบทที่ 5 เรื่อง Exploiting Formatted Output Functions

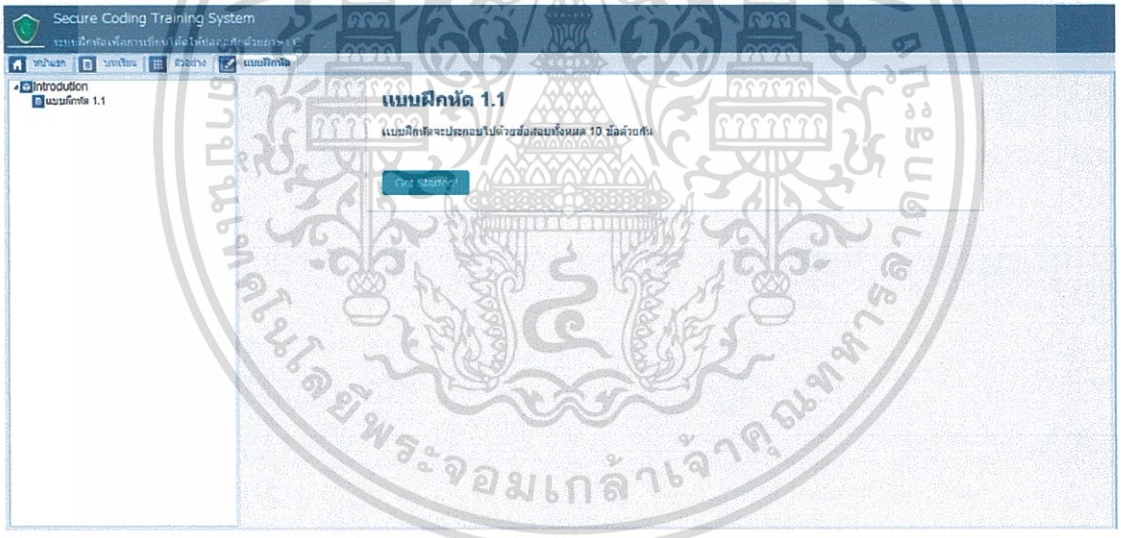
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูป 4.20 เว็บไซต์หน้าของการเรียนการสอนบทที่ 5 เรื่อง Mitigation Strategies

4.1.1 ทดลองใช้เว็บในส่วนของแบบทดสอบ

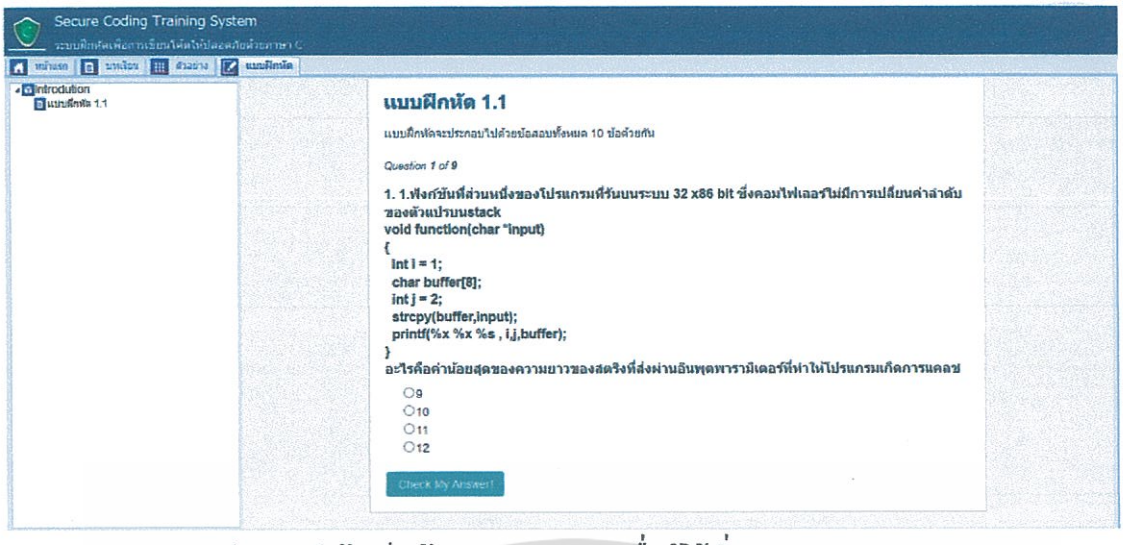
เมื่อผู้ใช้คลิกเข้าสู่แบบทดสอบของเว็บไซต์



รูป 4.21 เว็บไซต์หน้าแรกของบททดสอบ

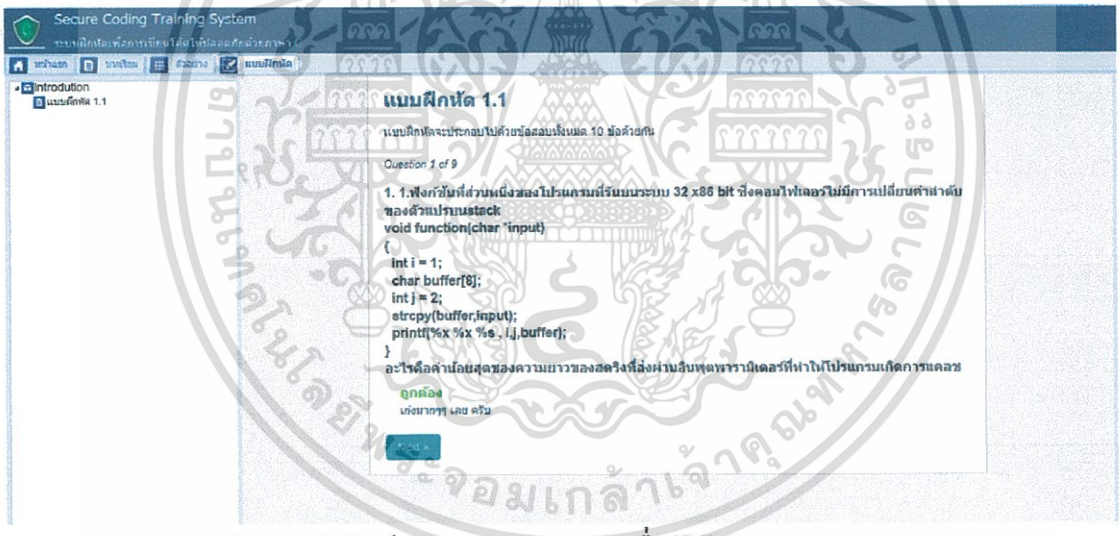
เมื่อผู้ใช้คลิกปุ่ม Get Started เพื่อเริ่มเข้าทำแบบทดสอบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



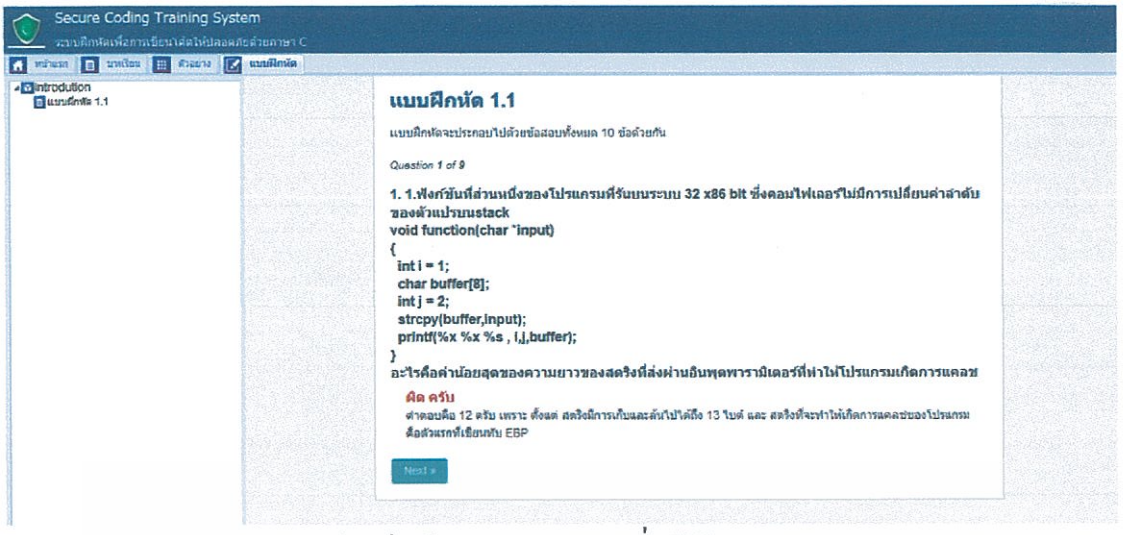
รูป 4.22 เว็บไซต์หน้าของบททดสอบเมื่อผู้ใช้เริ่มทำแบบทดสอบ

เมื่อผู้ใช้เลือกคำตอบได้แล้วให้คลิก Check My Answer จะทำการเช็คว่าคำตอบถูกหรือผิดถ้าผิดจะมีคำเฉลยให้ผู้ใช้ได้อ่านเพื่อทำความเข้าใจว่าผู้ใช้ผิดตรงไหน



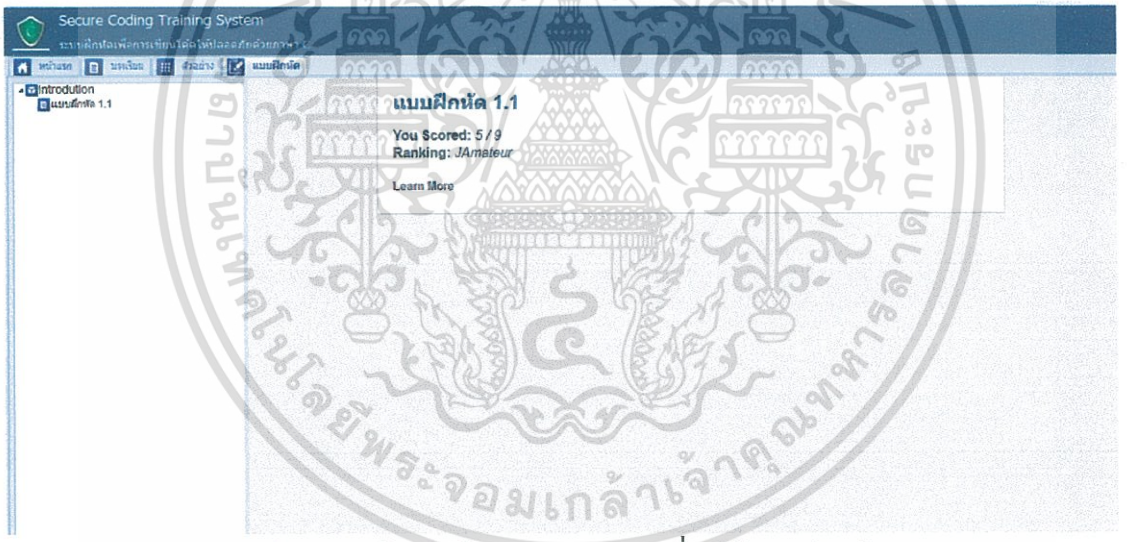
รูป 4.23 เว็บไซต์หน้าของบททดสอบเมื่อผู้ใช้ทำแบบทดสอบถูก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูป 4.24 เว็บไซต์หน้าของบททดสอบเมื่อผู้ใช้ทำแบบทดสอบผิด

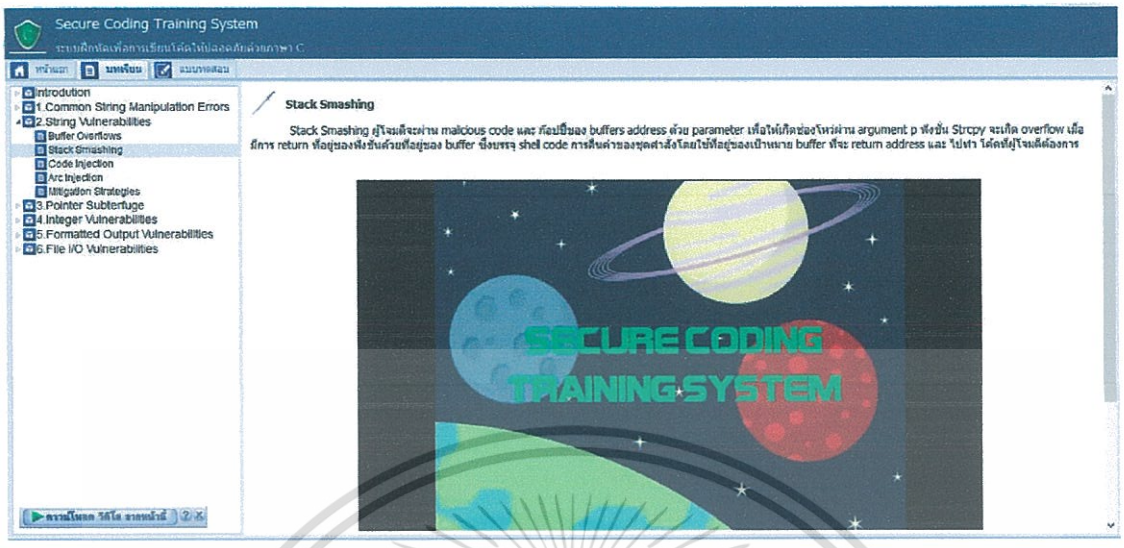
เมื่อผู้ใช้ทำแบบทดสอบข้อนั้นเสร็จให้คลิกปุ่ม Next เพื่อทำข้อต่อไปจนถึงข้อสุดท้ายเว็บไซต์จะทำการสรุปผลคะแนนและจัดลำดับความรู้ของผู้ใช้



รูป 4.25 เว็บไซต์หน้าของบททดสอบเมื่อผู้ใช้ทำเสร็จแล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

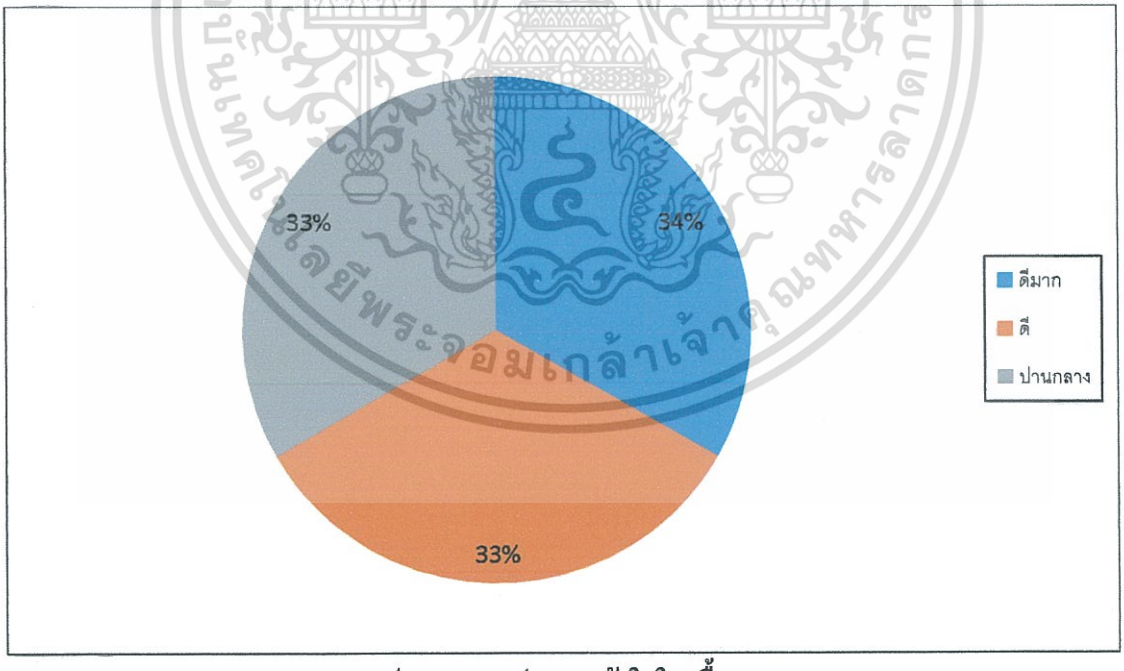
4.1.2 ทดลองใช้เว็บในส่วนของวีดีโอประกอบการสอน



รูป 4.26 เว็บไซต์ส่วนของวีดีโอประกอบการสอน

4.2 การประเมินความพึงพอใจในการใช้เว็บ

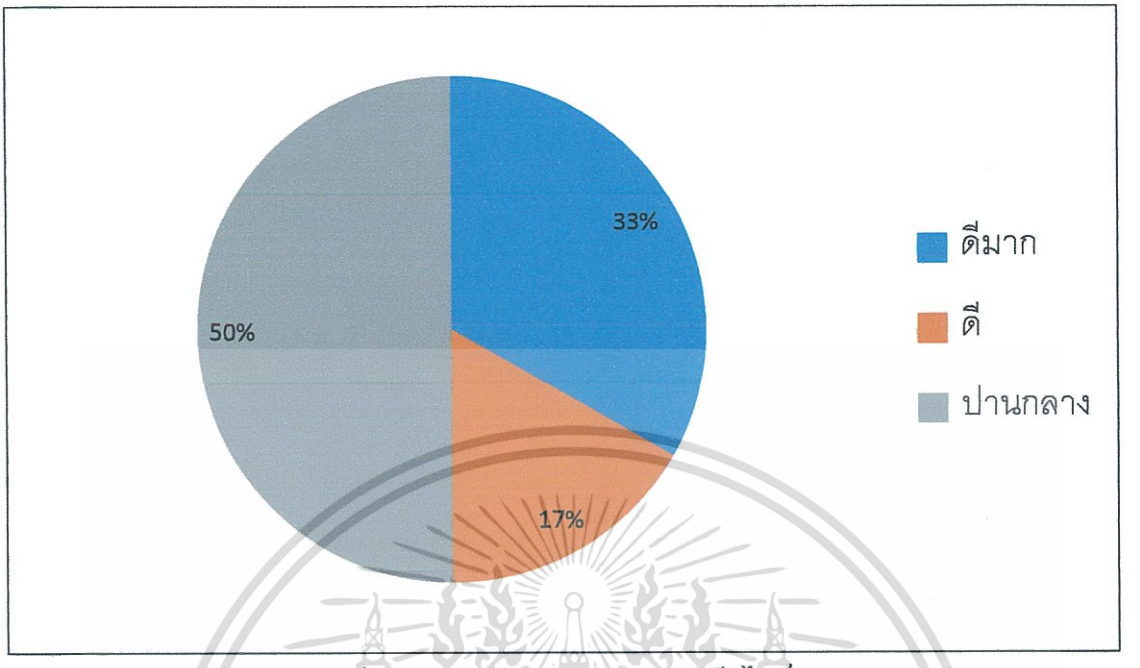
ความเข้าใจในเนื้อหา



รูป 4.27 กราฟความเข้าใจในเนื้อหา

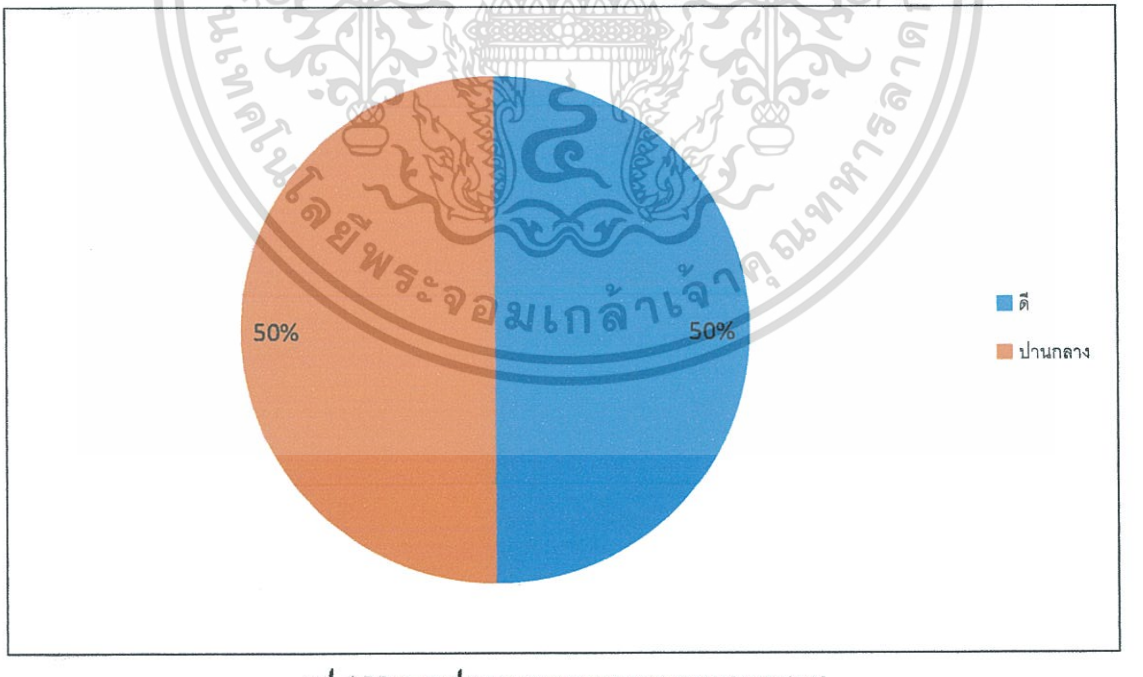
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ความสวยงามของเว็บไซต์



รูป 4.28 กราฟความสวยงามของเว็บไซต์

ความเหมาะสมของแบบทดสอบ



รูป 4.29 กราฟความเหมาะสมของแบบทดสอบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
บทสรุป

5.1 สรุปผลการทดลอง

ปัญหาช่องโหว่ เป็นปัญหาพื้นฐานของระบบคอมพิวเตอร์ตั้งแต่เรามีคอมพิวเตอร์จนถึงปัจจุบัน ยิ่งมีคอมพิวเตอร์ต่อไว้กับ internet มากขึ้นเพียงใด ก็มีเป้านิ่งให้ hackers ทั้งหลายได้โจมตีมากขึ้น แม้ปัจจุบันจะมีแนวทางที่เสนอออกมามากมายแต่ก็ยังไม่มีแนวทางใดที่สมบูรณ์แบบ ทางแก้ปัญหาที่ควรทำเป็นขั้นต้นคือฝึกให้นักพัฒนาซอฟต์แวร์ทั้งหลายใส่ใจกับปัญหาและทราบวิธีการรับมือ

5.2 ปัญหา และ อุปสรรค

1) เนื้อหาด้าน secure codeing c หาค่อนข้างยาก
2) มีปัญหาด้านการแปลภาษาอังกฤษเพราะเนื้อหาส่วนใหญ่จะเป็นภาษาอังกฤษทำให้ใช้เวลามากในการแปลเนื้อหา
3) ตัวอย่างและวิธีการทางด้าน secure c นั้นมีคนศึกษาน้อยทำให้การหาข้อมูลนั้นทำได้ยาก
4) มีปัญหาด้านการนำเสนอแบบเรียนให้น่าสนใจ

5.3 แนวทางการพัฒนาต่อ

1) ปรับปรุงความสวยงามของหน้าเว็บไซต์ให้มีความหลากหลายเพื่อเพิ่มการดึงดูดคนเข้าใช้เว็บไซต์
2) เพิ่มจำนวนแบบฝึกหัดเพื่อให้ผู้ใช้ได้ฝึกมากขึ้น
3) พัฒนาเพิ่มเติมภาษาอื่นๆ นอกจากภาษา c

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บรรณานุกรม

Robert C. Seacord. 2005. **Secure Coding in C and C++** . Addison Wesley Professional.

นาย กิจชัย รังสิมันต์ไพบูลย์ และนาย ดนวัต กัณฑ์สุข. 2548. "**ระบบต่อต้านการบุกรุกช่องโหว่ทางซอฟต์แวร์**"วิทยานิพนธ์ วิศวกรรมศาสตร์บัณฑิต สาขาวิศวกรรมคอมพิวเตอร์สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้