## 2 – Лаборатория иши

## Хотиранинг тўлиб тошиш таҳдиди

**Ишдан мақсад:** C, C++ дастурлаш тилида хотиранинг тўлиб тошиш таҳдиди билан танишиб чиқиш.

### Назарий қисм

Амалда кўп учрайдиган дастурлаш тилларидаги камчиликлар одатда, тақиқланган форматдаги ёки ҳажмдаги маълумотлар киритилиши натижасида келиб чиқади. Бу турдаги таҳдидлар ичида кенг тарқалгани бу – хотиранинг тўлиб тошиш таҳдиди саналади.

Масалан, веб сайтда фойдаланувчидан маълумотлар киритилиши талаб этилса (исми, фамиляси, йили, ва ҳак.), фойдаланувчи томонидан киритилган “исм” майдонидаги маълумот сервердаги N та белги ҳажмига эга соҳага ёзилади. Агар киритилган маълумот узунлиги N дан катта бўлган ҳолда, хотиранинг тўлиб тошиши ҳодисаси юзага келади.

Агар бузғунчи томонидан “керакли” маълумот киритилса, бу ўз навбатида компьютерни бузулишига олиб келади.

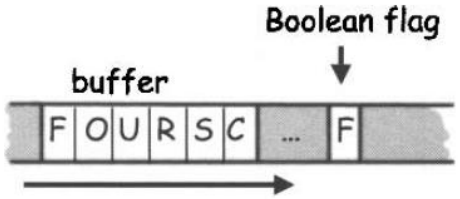
Қуйида C дастурлаш тилида ёзилган код келтирилган бўлиб, агар бу код компиляция қилинса хотиранинг тўлиб тошиши ҳодисаси келиб чиқади.

```
int main()
    {
    int buffer [10];
       buffer [20] =37;
       }
```

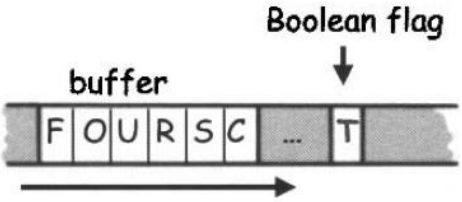
Сабаби 10 байт ўлчамдаги хотиранинг 20 байтига маълумот ёзилмоқда. Бу эса хотиранинг рухсат этилмаган манзилига мурожаатни келтириб чиқаради.

Агар дастурий маҳсулот аутентификацияни таъминлаш мақсадида яратилган бўлиб, аутентификация натижаси бир бит билан ифодаланади. Агар хотиранинг тўлиб тошиши натижасида ушбу бит бузғунчи томонидан муофақиятли ўзгартирилса Триди ўзини Алиса деб таништириш имкониятига

эга бўлади. Бу ҳолат қуйидаги 2.1-расмда келтирилган. Бу ерда F аутентификациядан муваффақиятли ўтилмаганлигини билдиради. Агар Триди F (0 ни) майдон қийматини T (1 га) ўзгартирса, дастурий таъминот Тридини Алиса сифатида танийди ва унга ресурсларидан фойдаланиш имкониятини яратади (2.2 - расм).

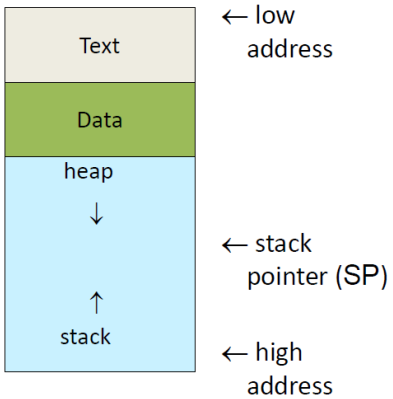



2.1 – расм. Хотира ва мантиқий байроқ




2.2 – расм. Содда хотирани тўлиб тошиши

Хотирани тўлиб тошиш ҳодисасини чиқурроқ ўрганишдан олдин замонавий компьютернинг хотира тузилиши билан танишилиб чиқилади. Компьютер хотирасининг соддалашган кўриниши қуйидаги 2.3 – расмда келтирилган.




2.3 – расм. Хотиранинг тузилиши

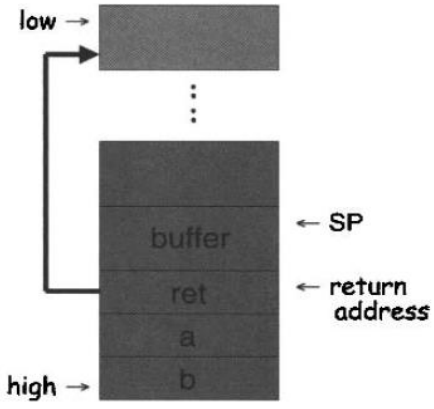
Бу ерда *text* мадонида кодлар сақланиб, *data* соҳасида статик катталиклар сақланади. *Heap* соҳаси динамик маълумотларга тегишли бўлиб, *stack* процессор учун «кераксиз қоғоз» вазифасини ўтайди. Масалан, динамик

локал ўзгарувчилар, функция параметлари, функцияларнинг қайтариш манзиллари каби маълумотлар stackда сақланади. *Stack pointer* ёки *SP* эса stackни энг юқорисини кўрсатади. Расмда stackни қуйидан юқорига чиқиши ҳолати билан ифодаланган.

**Stackни аварияга учратиш.** Stackни аварияга учраш ҳодисаси асосан хотирани тўлиб тошиши натижасида келиб чиқади. Бу турдаги таҳдидда Триди функцияларни чақирилиши давомида stackни текширади. Функцияни чақириш давомида stackдан фойдаланиш тартиби қуйидаги кодда келтирилган.

```
void func(int a, int b)
{
        char buffer[10];
}
void main()
{
        func(l,2);
        }
```

Қачонки *func* функцияси чақирилганда функциянинг параметрлари stack да итариб чиқарилади (2.4 – расм).

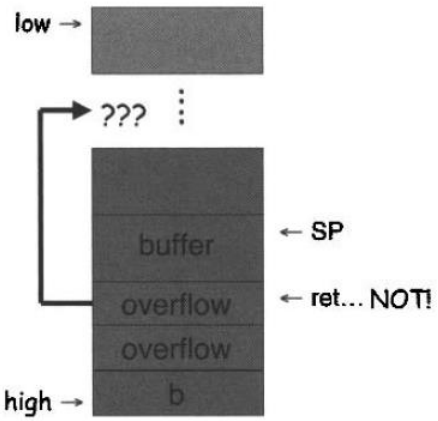



2.4 – расм. Stackга мисол

Бу ерда stack функцияни бажарилиши давомида *buffer* массивини яратиш учун фойдаланилмоқда. Бундан ташқари stack функцияни қайтарувчи, функция бажарилиб бўлинганидан кейин ўтиши керак бўлган манзилини ҳам ўзида сақлайди. Расмда кўрсатилгани каби *buffer* қайтувчи манзилдан (ret) дан юқорида жойлашган, яъни, қайтарулувчи манзилдан сўнг *buffer* stackда

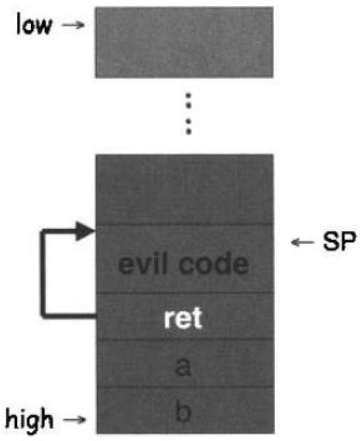
юкланади. Натижада, агар хотирани тўлиб тошиши юзага келса, у ҳолда хотиранинг *ret* соҳаси қайтадан ёзилади. Бу таҳдид натижасида олиниши мумкин бўлган, реал натижа.

Агар Триди томонидан хотира тўлдирилса ва қайтарувчи манзил тасодифий битлар билан тўлдирилса, у ҳолда дастур мавжуд бўлмаган манзилга сакрайди ва тизим аврияга учрайди (2.5-расм).




2.5 – расм. Хотиранинг тўлиб тошиш муаммоси

Бу ҳолда дастур ишини тўхтатгандан Триди хурсанд бўлиши аниқ. Агар Триди янада ақллироқ бўлса ва буферни тасодифий битлар билан эмас, балки муҳим хотира манзили билан тўлдирса ва бу хотира манзилига бирор зарарли дастур бўлса, у ҳолда жиддийроқ муаммо бўлиши аниқ (2.6 - расм).

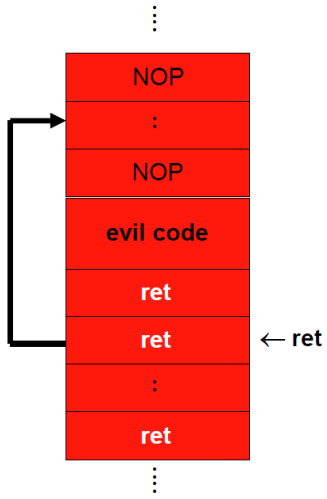



2.6 – расм. Зарарли кодни юклаш

Бу ҳолда Триди қуйидаги икки муаммога дуч келиши мумкин. Биринчиси, Триди зараркунанда дустурни хотиранинг қайси манзилига

ёзилганини билмайди. Иккинчиси эса, stackда функцияни қайтувчи манзилини аниқ билмайди.

Қуйидаги икки содда хийла натижасида, хотирани тўлиб тошиш жараёнини тезлаштириш мумкин. Биринчиси бу, зараркунанда дастур кодини хотира бўшлиқлари билан (NOP) тўлдириш бўлса, иккинчиси эса, исталган такрорланувчи қайтувчи манзилни қўйишдир (2.7 - расм).




2.7 – расм. Хотирани NOP билан тўлдириш

Бу таҳдид одатда кўплаб, дастурий маҳсулот учун керакли бўлган сериал калитларни бузишда кенг фойдаланилади.

## Амалий қисм

1. Биринчи мисолда *buf* массивига ўлчамдан катта бўлган қийматни ёзинг.

```
#include <iostream>
int main() {
    char buf[8];
    std::cout << "Please enter INPUT:" << std::endl;
    gets(buf);
    std::cout<<"Your input: "<<buf<<std::endl;
    return 0;
}
```

Натижа:

```
Please enter INPUT:
hello world
Your input: hello world
```

2. Юқоридаги мисолни хавфсиз шаклда ёзилган шакли:

```
#include <iostream>
int main() {
    char buf[8];
    std::cout << "Please enter INPUT:" << std::endl;
    fgets(buf,8,stdin);
    std::cout<<"Your input: "<<buf<<std::endl;
    return 0;
}
```

Натижа:

```
Please enter INPUT:
Hello world
Your input: Hello w
```

3. Юқоридаги каби қуйидаги функциялар билан натижалар олинг:
   - ✓ `strcpy() -> strncpy() – контентни буферга кўчириш.`
   - ✓ `strcat() -> strncat() – буферларни бирлаштириш.`
   - ✓ `sprintf() -> snprintf() – буферни тўлдириш.`
4. Стекни ҳимоялашда одатда *Canary* дан фойдаланилади. Ушбу технологияни ишлашини тушунтиринг.
5. *Ихтиёрий ҳолда қуйидаги манзилда келтирилган манзилдаги маълумотлар билан танишиб чиқиб, уни амалга оширишга ҳаракат қилинг.*
   a. http://www.cse.scu.edu/~tschwarz/COEN252_09/Lectures/BufferOverflow.html
   b. http://resources.infosecinstitute.com/buffer-overflow-attack-defense/