

# Компютърна сигурност

проф. д-р инж. Христо Вълчанов

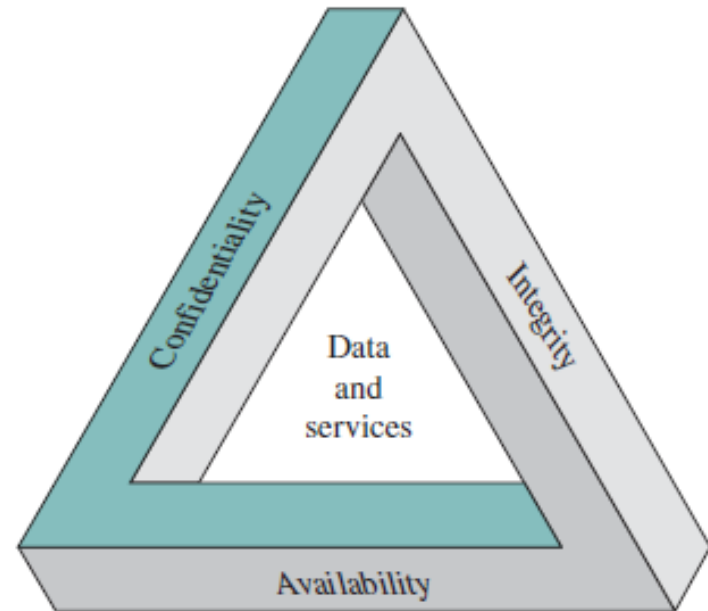
<http://cs.tu-varna.bg>

# Информационна сигурност

---

Цели:

- Конфиденциалност
- Интегритет
- Наличност



# Стандарти за информационна сигурност

---

- ISO 7498-2 SecurityArchitecture - OSI защитена архитектура;
- ISO 9594-8 - насоки за автентификация (X.509);
- ISO 10181 - OSI насоки за защита на информацията;
- ISO/IEC 27002 - за управление на сигурността.

# Заплахи за сигурността

---

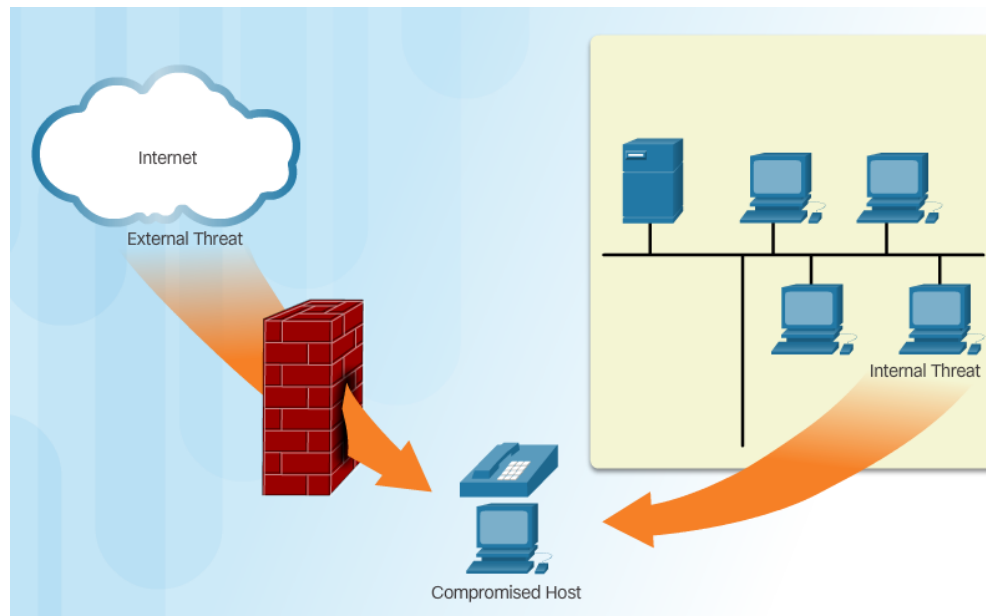
Това е потенциална опасност за:

- информацията
- компютърната система
- мрежовата услуга

# Видове заплахи

По произход са:

- Вътрешни
- Външни



# Видове заплахи

---

По начин на организиране са:

- Структурирани – предварително прецизно планирани
- Неструктурирани – не са планирани

# Видове заплахи (3)

---

По начин на провеждане на атаката са:

- Пасивни – снифер на пароли, анализ на трафика
- Активни – опит за логване, потискане на услугата (DoS), маскарадинг, модификация на съобщения

# Еволюция на заплахите

---

С годините атаките към мрежовата сигурност се развиват.

- Към 1985 атакуващият трябва да е много добре запознат с компютъра, програмирането, да има знания по мрежи, да познава детайлите на средствата, с които ще атакува.
- Средствата, с които се атакува се развиват и вече не изискват високи знания в компютърната област.
- Хората, които преди не са правили компютърни престъпления поради незнание в ИТ областта, вече могат да ги правят.



# Вътрешни атаки

---

**Backdoor** – секретна входна точка в програмен код чрез която може да се осигури неоторизиран достъп до системата.

# Вътрешни атаки

---

**Logical Bomb** – вграден код в легитимна програма по време на разработката от разработчика. Изпълняват се нерегламентирани действия при настъпване на някакво събитие: настъпване на дата, определен резултат и др.

# Вътрешни атаки

---

**Trap Doors** – вграден код в системата, който игнорира нормална проверка за логване.

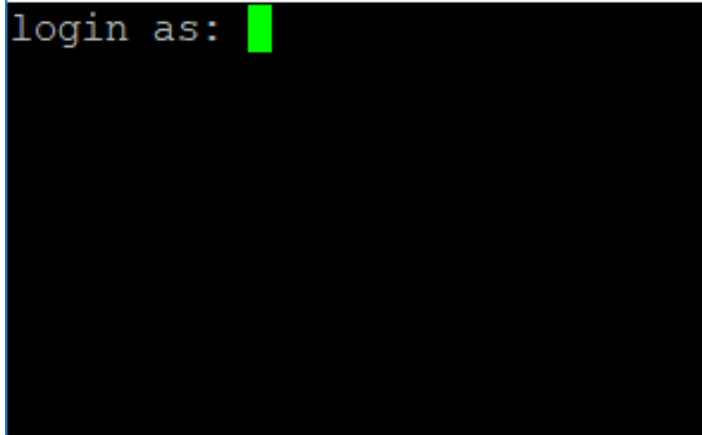
```
while (1) {  
    printf("login:");  
    get_string(name);  
    disable_echoing();  
    printf("password:");  
    get_string(password);  
    enable_echoing();  
    v=check_validity(name,password);  
    if (v) break;  
}  
execute_shell(name);
```

```
while (1) {  
    printf("login:");  
    get_string(name);  
    disable_echoing();  
    printf("password:");  
    get_string(password);  
    enable_echoing();  
    v=check_validity(name,password);  
    if (v || strcmp(name,"DOOR")==1)  
        break;  
}  
execute_shell(name);
```

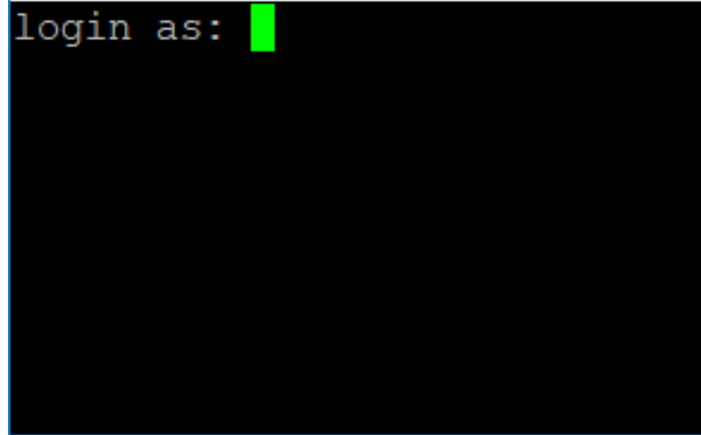
# Вътрешни атаки

---

**Login Spoofing** – събиране на пароли без знанието на потребителите.

A terminal window with a black background. The text "login as:" is displayed in a light blue monospace font. A red cursor is positioned at the end of the text.

```
login as: 
```

A terminal window with a black background. The text "login as:" is displayed in a light blue monospace font. A red cursor is positioned at the end of the text.

```
login as: 
```

# Използване на бъгове в кода

---

**Buffer Overflow Attack** – достъп до невалидно адресно пространство.

```
int i;  
char c[1024];  
i = 12000;  
c[i] = 0;
```

# Използване на бъгове в кода

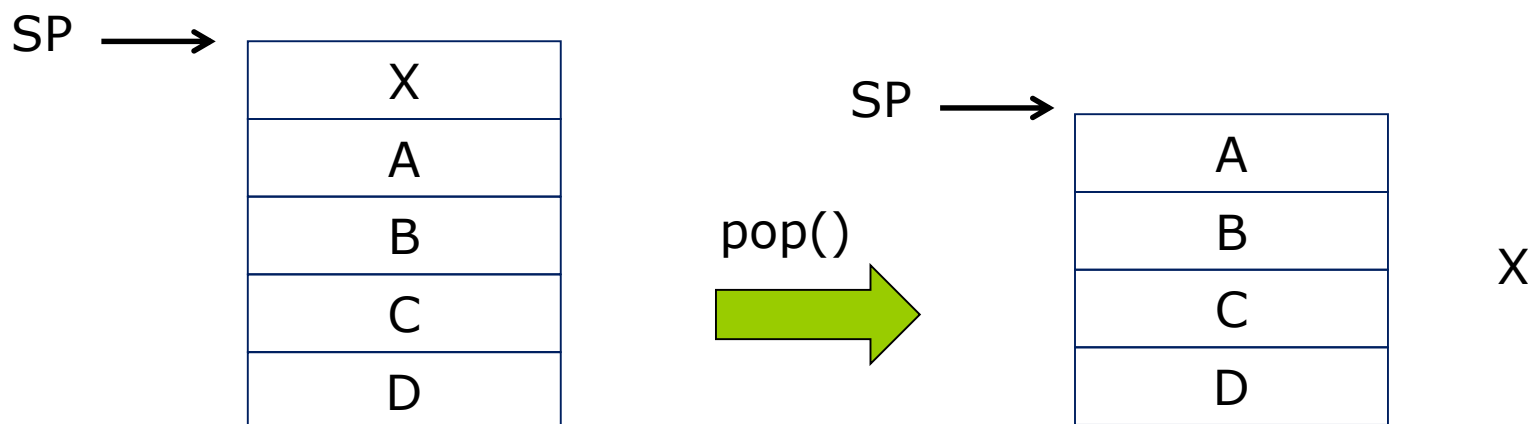
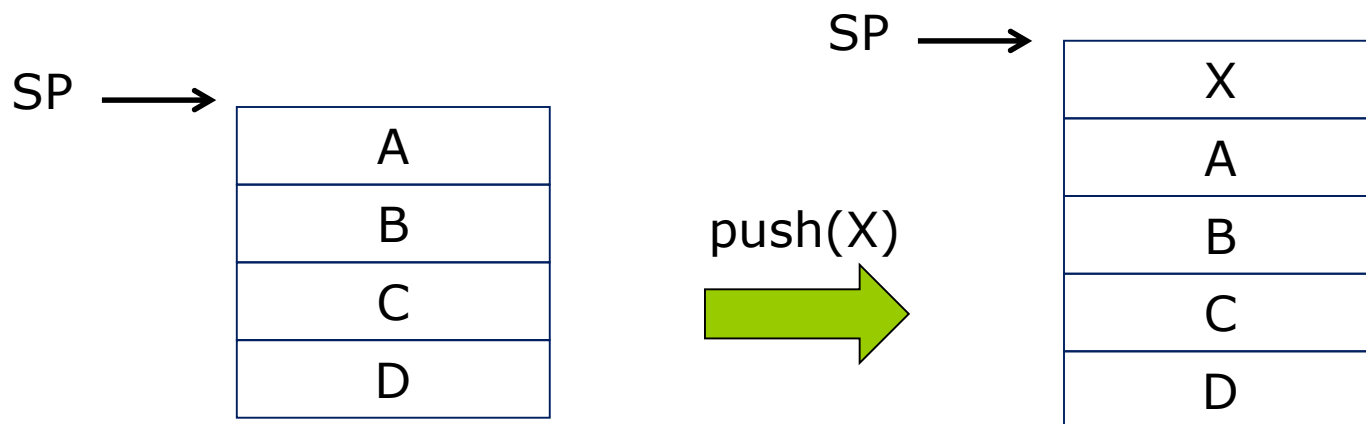
---

```
#include <stdio.h>

int main(int argc, char **argv) {
    char buf[8]; // buffer for eight characters

    gets(buf); // read from stdio (sensitive function!)
    printf("%s\n", buf); // print out data stored in buf
    return 0; // 0 as return value
}
```

# Buffer overflow – използване на стека



# Buffer overflow – използване на стека

```
void foo(const char* input) {  
    char buf[10];  
    printf("Hello World\n");  
}  
  
int main(int argc, char* argv[])  
{  
    foo(argv[1]);  
    return 0;  
}
```

Stack преди извикване на  
foo

buf[3]	buf[2]	buf[1]	buf[0]
buf[7]	buf[6]	buf[5]	buf[4]
		buf[9]	buf[8]
argument of foo			
return address			
Stack след извикване на foo			



# Buffer overflow – използване на стека

---

```
void foo(const char* input) {
    char buf[10];
    printf("My stack looks like:\n%p\n%p\n%p\n%p\n%p\n%p\n\n");
    strcpy(buf, input);
    printf("%s\n", buf);
    printf("Now the stack looks like:\n%p\n%p\n%p\n%p\n%p\n%p\n\n");
}

void bar(void) {
    printf("Augh! I've been hacked!\n");
}

int main(int argc, char* argv[]) {
    printf("Address of foo = %p\n", foo);
    printf("Address of bar = %p\n", bar);
    if (argc != 2) {
        printf("Please supply a string as an argument!\n");
        return -1;
    }
    foo(argv[1]);
    return 0;
}
```

# Buffer overflow – използване на стека

---

```
C:>stackover.exe Hello
Address of foo = 00401000
Address of bar = 00401050
My stack looks like:
00000000
00000A28
7FFDF000
0012FEE4
004010BB
0032154D
```

```
Hello
Now the stack looks like:
6C6C6548
0000006F
7FFDF000
0012FEE4
004010BB
0032154D
```

# Buffer overflow – използване на стека

---

```
C:>stackover.exe ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

```
Address of foo = 00401000
```

```
Address of bar = 00401050
```

```
My stack looks like:
```

```
00000000
```

```
00000A28
```

```
7FFDF000
```

```
0012FEE4
```

```
004010BB
```

```
0032154D
```

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

```
Now the stack looks like:
```

```
44434241
```

```
48474645
```

```
4C4B4A49
```

```
504F4E4D
```

```
54535251
```

```
58575655
```

**RUN-TIME ERROR !**

# Buffer overflow – използване на стека

---

➤ **Подмяна на return address с**

**00 40 10 50**

# Buffer overflow – използване на стека

---

**ABCDEFGHJKLMNOP\x50\x10\x40**

# Buffer overflow – използване на стека

---

```
C:>stackover.exe ABDEFGHIJKLMNOPxxxxxx
```

```
Address of foo = 00401000
```

```
Address of bar = 00401050
```

```
My stack looks like:
```

```
00000000
```

```
00000A28
```

```
7FFDF000
```

```
0012FEE4
```

```
004010BB
```

```
00321599
```

```
ABDEFGHIJKLMNOPPP>@
```

```
Now the stack looks like:
```

```
44434241
```

```
48474645
```

```
4C4B4A49
```

```
504F4E4D
```

```
00401050
```

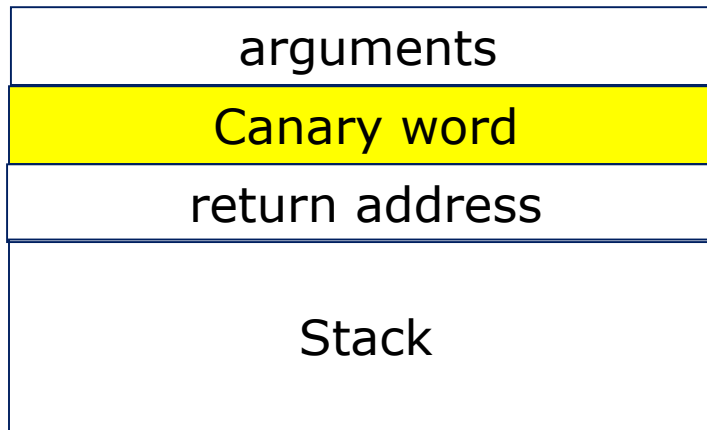


```
00321599
```

```
Augh! I've been hacked!
```

# Buffer overflow – защита

---



# Зловреден софтуер

---

**Trojan Horse** – софтуер, който има някаква официална полезна функционалност, но в чийто код има прикрит злонамерен код.



# Зловреден софтуер

---

**Mobile Code** – програми, скриптове, макроси, които се прехвърлят от отдалечена система на локалната и се изпълняват без явна намеса на потребител. Действат като механизъм за троянски коне (Java applets, JavaScript, VBScript)

# Вируси

---

Злонамерен софтуер, който може да “инфектира” други програми чрез тяхната модификация. Включва се допълнителен код, който създава копия на вируса за заразяване на други програми. Може да извършва зловредни действия като изтриване на данни, криптиране и др.

# Видове вируси

---

**Boot sector virus** – инфектира master boot записа и се разпространява при стартиране на системата.

**Macro virus** - инфектират документи (Microsoft Office приложения).

**Polymorphic virus** – променя се при всяка инфекция, затруднявайки разпознаването му.

# Червеи

---

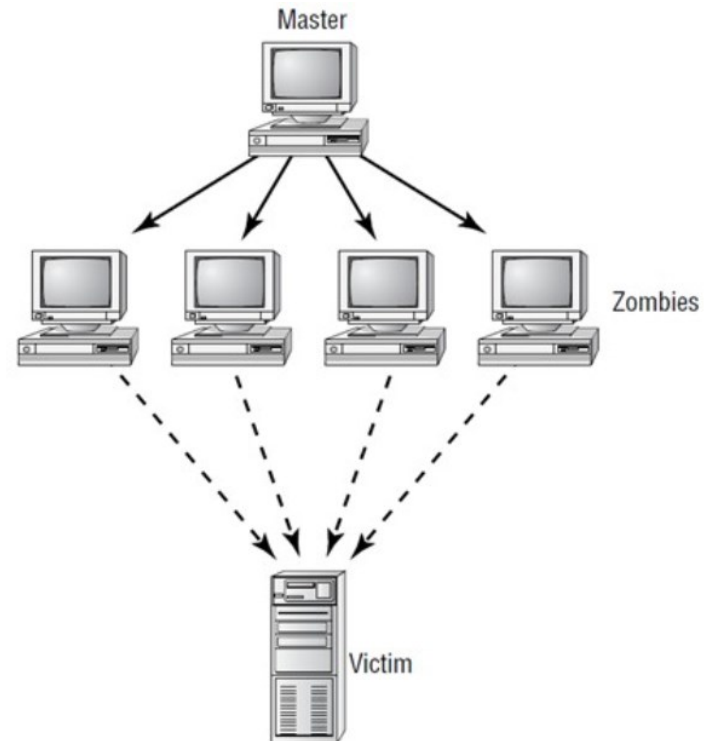
Програма, която се саморепликира и изпраща копия към други компютри през мрежата. В допълнение към разпространението може да изпълнява зловредни действия (имплантиране на троянски коне, повреждане на информацията).

# Ботнет

Мрежа от заразени машини  
- зомбита.

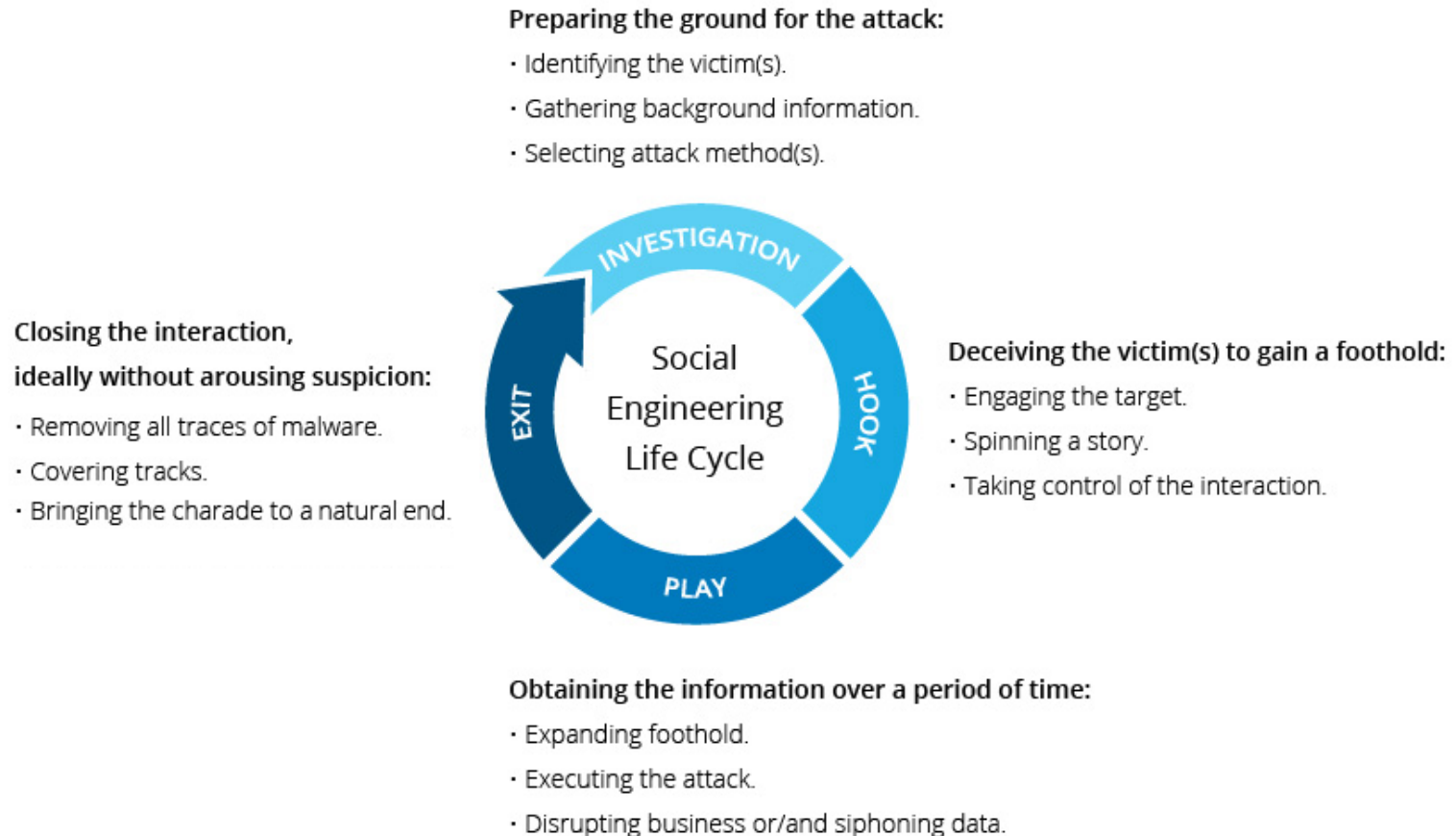
Зомбитата се контролират  
от манипулатор – мастър  
бот.

Реализират основно  
Distributed Denial of Service  
атака (DDoS)



# Social Engineering

---



Използване на психологическа манипулация на потребителите за осъществяване на пробив в сигурността

---

# Принципи на защита и сигурност

---

**Принцип на най-малките привилегии –**  
на програмите, потребителите и  
системите трябва да бъдат дадени  
единствено привилегии, достатъчни за  
изпълнение на техните задачи.

# Техники за компютърна сигурност

---

- Автентикация
- Контрол на достъп
- Разпознаване на заплахи
- Защита от зловреден софтуер



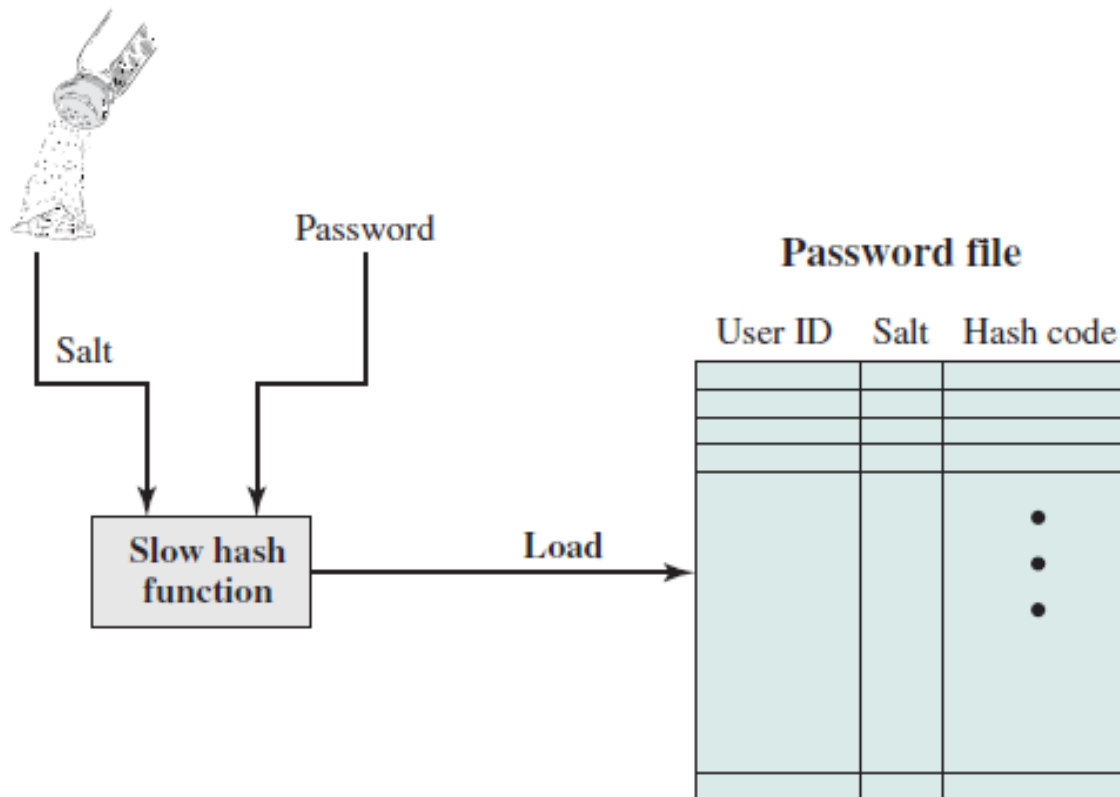
# Автентикация, базирана на пароли

---

- Паролата се хешира с псевдослучайно число – *salt*.
- Типично за Linux се използва MD5 хеш алгоритъм и 48 бита *salt* стойност.

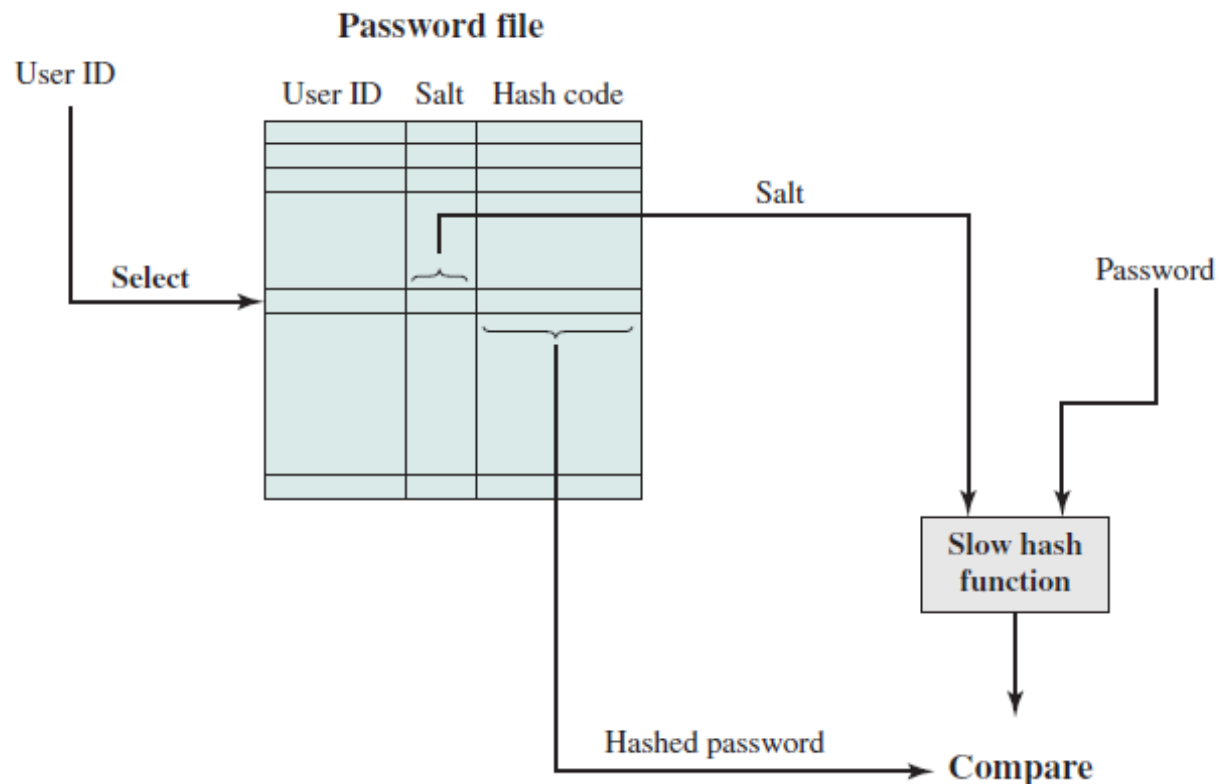
# Автентикация, базирана на пароли (2)

## Въвеждане на нова парола



# Автентикация, базирана на пароли (2)

## Проверка на парола



# Автентикация, базирана на токени

---

- **Memory card** – карти с памет, съхраняващи код за достъп.
- **Smart card** – малко устройства с вграден микропроцесор, памет и интерфейс и поддържат протоколи за автентикация.
- **Биометрична автентикация** – персонална автентикация на базата на уникални физически характеристики на потребител.

# Pluggable Authentication Modules

---

- PAM - пакет от споделени библиотеки, даващи възможност за избор на автентикация на потребители.
- Въвежда Middleware слой между приложението и автентикационния механизъм.
- Може да поддържа данни за сесията.
- Разширява възможностите на Linux да бъде от класа на enterprise OS.

# Контрол на достъпа

---

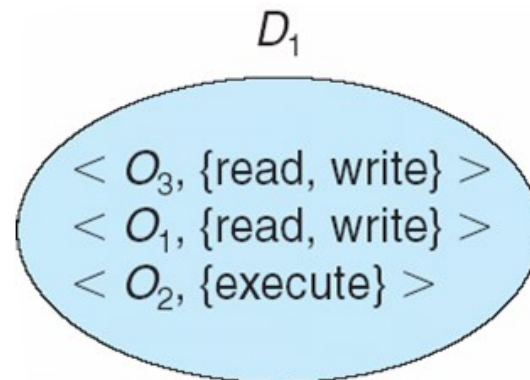
Политиките за контрол на достъпа указват какви типове на достъп за разрешени, при какви условия и за кого:

- **Discretionary access control (DAC)** – контрол базиран на идентичността на заявителя и на правила на негов достъп. Може да се предоставят права на други субекти.
- **Role-based access control (RBAC)** – контрол базиран на ролята на потребителите в системата и на права на достъп за съответната роля.

# Discretionary access control (DAC)

---

- Компютърната система се разглежда като съвкупност от процеси и обекти
- Процесите оперират в защитена област (домейн), описваща до кои ресурси има достъп
- Възможността за изпълнение на операция върху обект- право на достъп
- Домейн – колекция от права на достъп, всяко във вида *<обект, множество права>*



# Реализация на домейн

---

Домейн може да бъде:

- Потребител. Възможните обекти за достъп зависят от идентичността на потребителя.
- Процес. Възможните обекти за достъп зависят от идентичността на процеса.
- Процедура. Възможните обекти за достъп съответстват на локалните променливи, дефинирани в процедурата.

В Linux домейн се асоциира с потребител.



# Матрица на достъп

Защитата се разглежда като матрица:

- Редовете представят домейните
- Колоните представят обектите
- **$Access(i, j)$**  е множество от операции, което процес, изпълняващ се в домейн  $i$ , може да изпълни върху обект  $j$ .

<div>object</div> <div>domain</div>	$F_1$	$F_2$	$F_3$	printer
$D_1$	read		read	
$D_2$				print
$D_3$		read	execute	
$D_4$	read write		read write	

# Използване на матрица на достъп

---

- Ако процес в домейн  $D_i$  се опита да изпълни операция **ор** върху обект  $O_j$ , то операцията **ор** трябва да бъде в матрицата за достъп
- Потребителят, създал обекта, може да дефинира колона за достъп до този обект

# Използване на матрица на достъп (2)

---

Матрицата за достъп разделя механизма на прилагане от политиката:

- Механизъм – операционната система предоставя матрицата и правата. Тя осигурява манипулирането с матрицата да бъде само от оторизирани компоненти.
- Политика – потребителят определя политиката. Кой и в какъв режим може да достъпва даден обект.

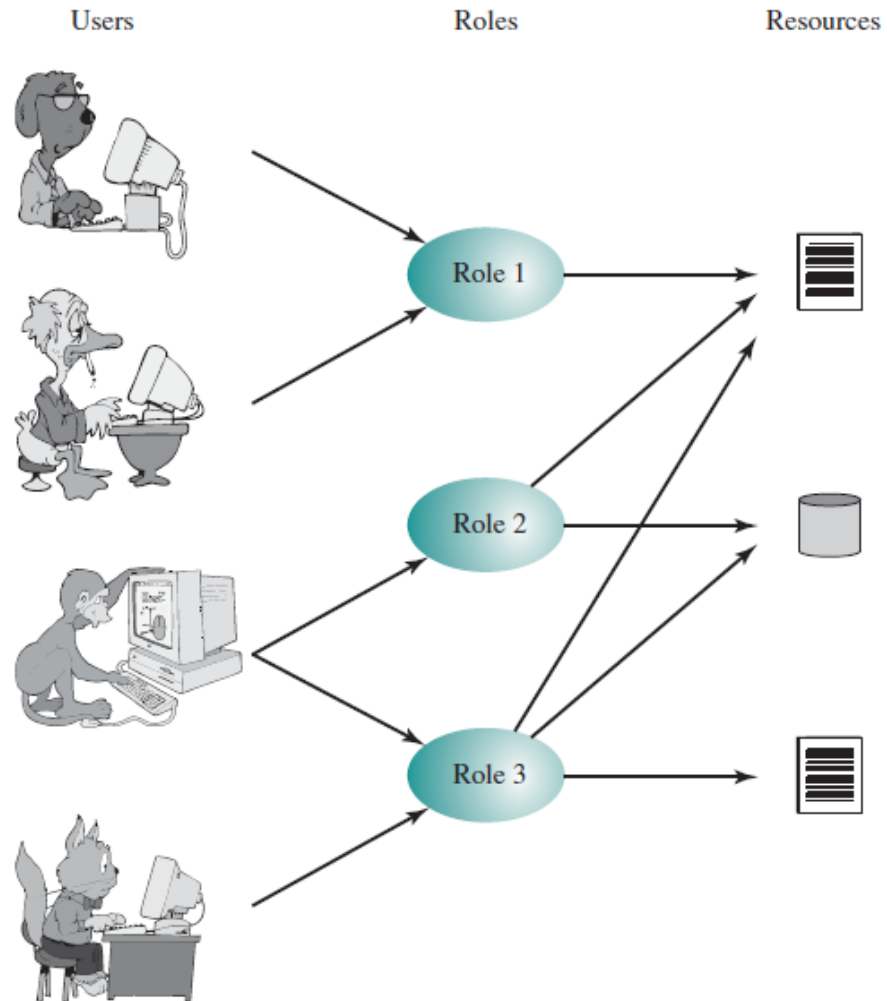
# Role-based access control (RBAC)

---

RBAC е контрол, базиран на роли, които потребителят може да заема в системата, вместо на неговата идентичност.

- Правата се присвояват на роли, вместо на индивидуални потребители.
- Потребителите се свързват към определени роли (най-често в зависимост от работата им в организацията).

# Role-based access control (2)



# Role-based access control (3)

Потребител може да заема множество роли и множество потребители могат да заемат една и съща роля:

	$R_1$	$R_2$	$\dots$	$R_n$
$U_1$	×			
$U_2$	×			
$U_3$		×		×
$U_4$				×
$U_5$				×
$U_6$				×
$\vdots$				
$U_m$	×			

# Role-based access control (4)

Всяка роля има специфични права на достъп до един или няколко ресурса. Роля може да се третира като обект – възможност за йерархии на роли.

		Objects								
		R <sub>1</sub>	R <sub>2</sub>	R <sub>n</sub>	F <sub>1</sub>	F <sub>1</sub>	P <sub>1</sub>	P <sub>2</sub>	D <sub>1</sub>	D <sub>2</sub>
Roles	R <sub>1</sub>	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
	R <sub>2</sub>		control		write *	execute			owner	seek *
	•									
	•									
	R <sub>n</sub>			control		write	stop			

# Модел на сигурност в Linux

---

- Системните извиквания са единственият начин за взаимодействие с ОС;
- Всеки потребител или негов процес се идентифицира с user-ID (UID).



# Модел на сигурност в Linux (2)

---

```
lp:x:4:7:lp:/var/spool/lpd:/bin/false
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/:/bin/false
news:x:9:13:news:/usr/lib/news:/bin/false
uucp:x:10:14:uucp:/var/spool/uucppublic:/bin/false
operator:x:11:0:operator:/root:/bin/bash
games:x:12:100:games:/usr/games:/bin/false
ftp:x:14:50:ftp:/home/ftp:/bin/false
smmsp:x:25:25:smmsp:/var/spool/clientmqueue:/bin/false
```

/etc/passwd

```
shutdown:*:9797:0:0:0:
halt:*:9797:0:0:0:
mail:*:9797:0:0:0:
news:*:9797:0:0:0:
uucp:*:9797:0:0:0:
operator:*:9797:0:0:0:
games:*:9797:0:0:0:
ftp:*:9797:0:0:0:
smmsp:*:9797:0:0:0:
mysql:*:9797:0:0:0:
rpc:*:9797:0:0:0:
sshd:*:9797:0:0:0:
```

/etc/shadow

# Контрол на достъп в Linux

---

- Режими на достъп: **read, write, execute**
- Три класа потребители в Unix / Linux:
  - **owner**
  - **group**
  - **public**
- За всеки файл и поддиректория има 9 защитни бита: 3 за owner, 3 за group, 3 за public.

# Модел на сигурност в Windows

---

- Security Reference Monitor – проверка на достъпа, генериране на лог файлове и манипулиране с права на достъп.
- Local Security Authority – прилагане на локалната политика за сигурност.
- Security Account Manager – локална база данни с информация за акаунтите. Използва се за локален достъп.

# Системи за откриване на заплахи

---

- Не може да се предотвратят всички атаки към системата
- Винаги ще има пропуски, нови атаки и нови атакуващи
- Търсят се начини за защита:
  - Повечето единични защиты могат да се провалят
  - Затова се прилага защита в дълбочина – на няколко слоя:
    - 1 слой - системи за откриване на проникване
    - 2 слой – системи за защита от проникване

# Допускания

---

- Действията в компютърните системи са наблюдаеми.
- И нормалната работа, и заплахите имат определена последователност от събития, която може да бъде проследена

# Наблюдение на системата

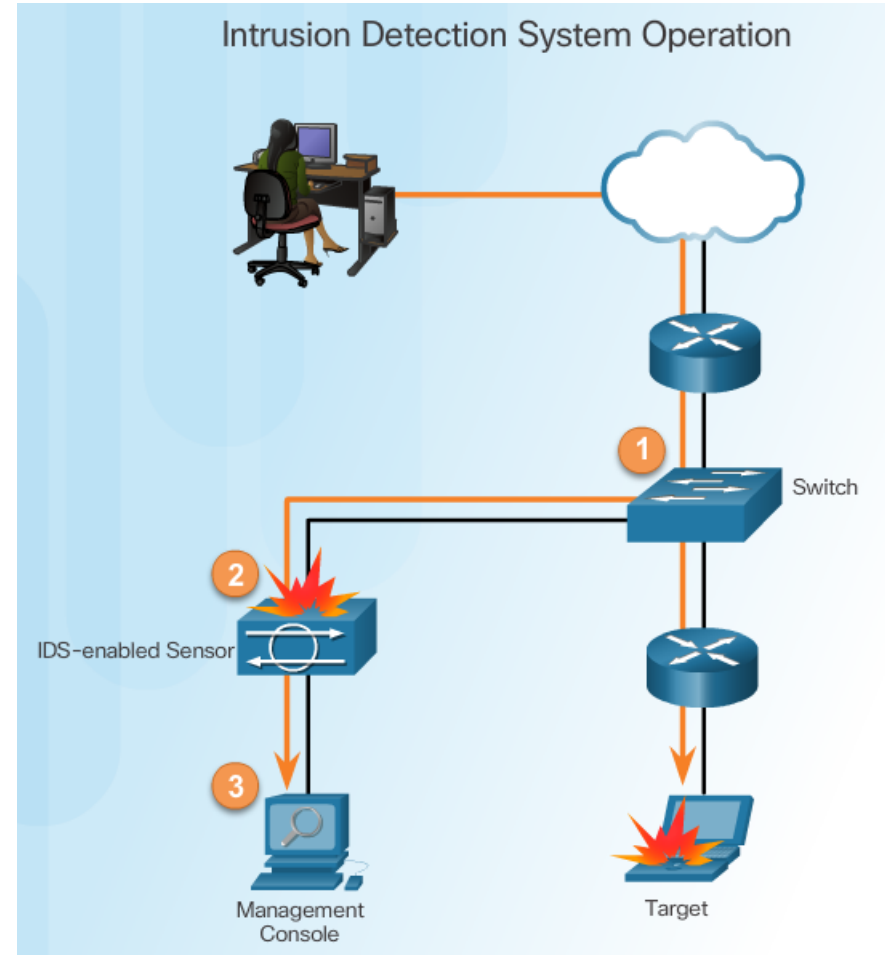
---

- Наблюдават се събитията, настъпващи в компютърна система или мрежа
- Анализират се признаци на възможни инциденти, които са нарушения или непосредствена заплаха от нарушение на правилата за компютърна сигурност
- Инцидентите са **злонамерени**, породени от:
  - Зловреден софтуер (червеи, шпионски софтуер...)
  - Неупълномощен достъп до системите от Интернет
  - Злоупотреба с привилегии на упълномощени потребители
  - Опити за получаване на допълнителни привилегии
- Инцидентите са и **случайни**:
  - Човек може да сгреша адреса на компютъра и случайно се опитва да се свърже с различна система без разрешение

# Разпознаване на атаки

Системи за разпознаване на атаки (Intrusion Detecting System - IDS):

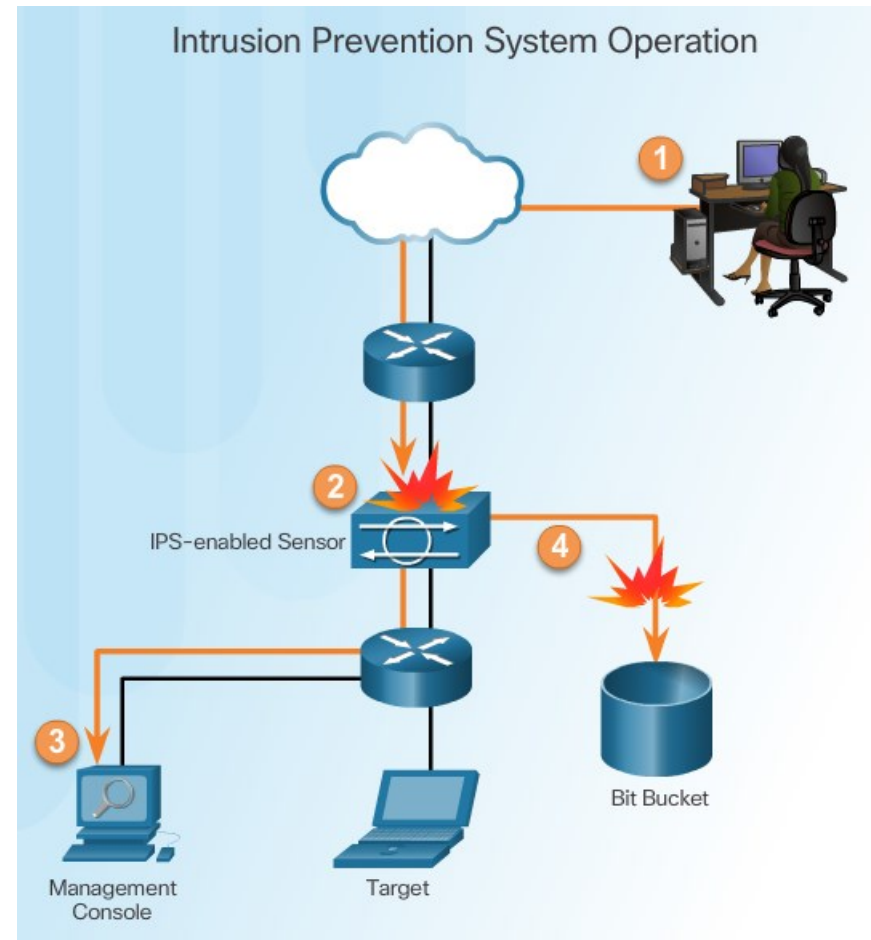
- Работят пасивно.
- Изискват трафика да бъде дублиран (mirror) през системата, за да бъде анализиран.



# Предпазване от атаки

Системи за предпазване от атаки (Intrusion Preventing System - IPS):

- Изпълняват се в режим in-line
- Наблюдават трафик на Layer 3 и Layer 4
- Могат да спрат един пакет от атаката да не достигне целта
- Реагират веднага и не позволяват на всеки злонамерен трафик да премине





# Подходи за разпознаване на атаки

---

- Базиран на аномалии – на основата на предварително събрани данни за нормално поведение на системата се дефинира граница. Настъпване на събития извън нея се счита за атака.
- Базиран на сигнатури – дефинира се множество правила или шаблони на атака. Трафикът се сравнява с тях.

# Защита от зловреден софтуер

---

- Обновяване на операционната система
- Премахване на уязвимости (patching)
- Антивирусен софтуер

**Въпроси?**