NTNU

Norwegian University of Science and Technology

Authorization and Multi-Level Security

Authentication and Single sign-on

Control hijacking attacks

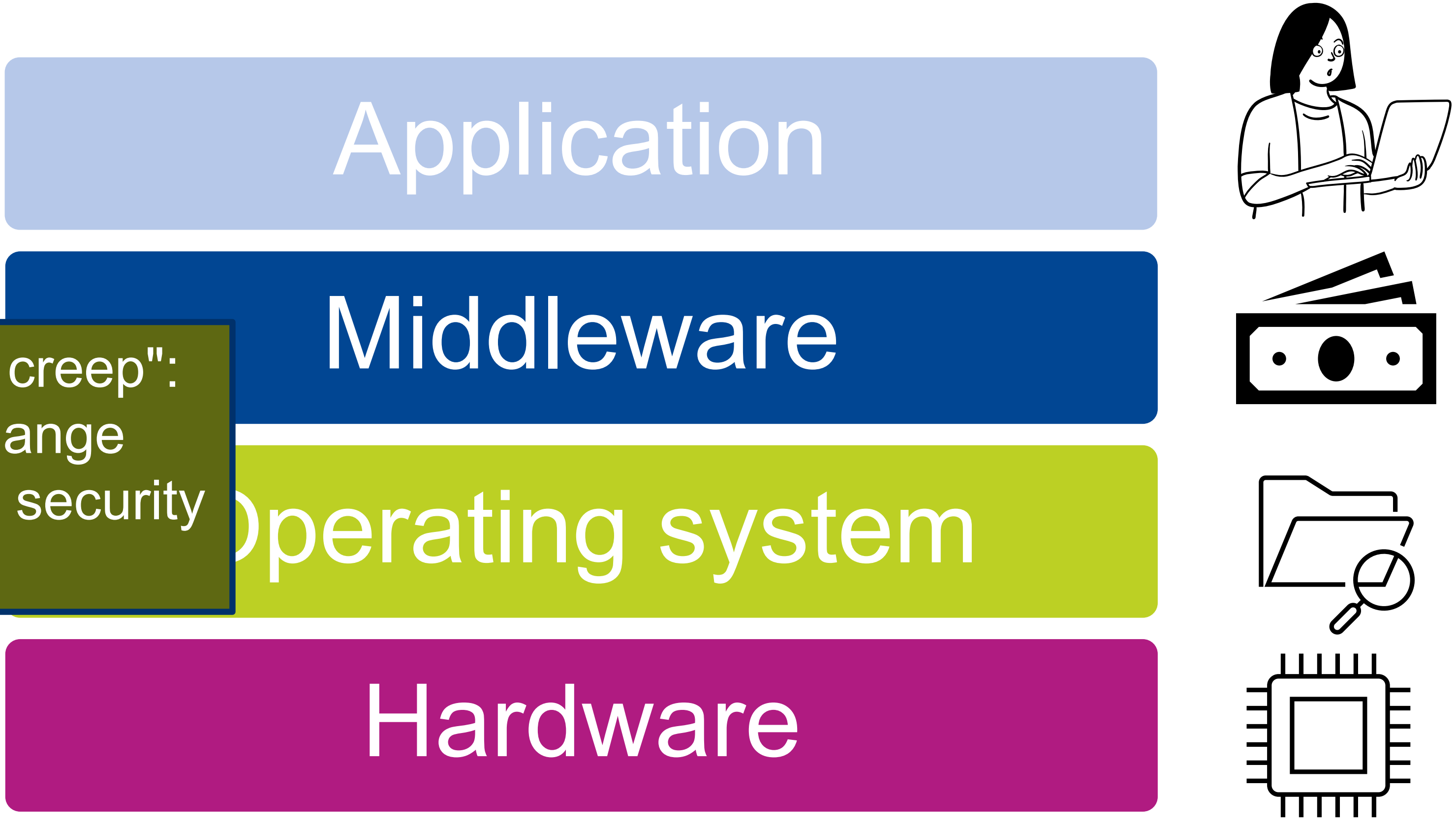
TDT4237 2025

Access Control

Policy	High-level rules, what is, and what is not, allowed
Model	Formal representation of the policy
Mechanism	Low-level implementation of the model
Awareness	Education
Management	Operation

"Privilege creep": People end up with more access than necessary

Access control on different levels



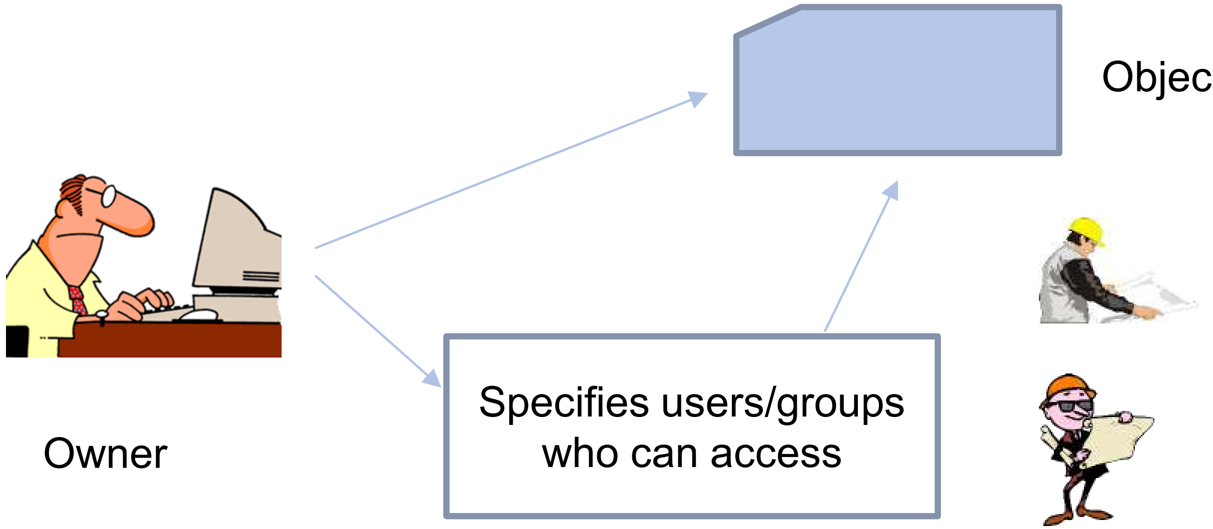
"Environmental creep": Environment change undermines the security model

Access control models

- Discretionary access control (DAC)
- Mandatory access control (MAC)
- Role-based access control (RBAC)
- Attribute-based access control (ABAC)
- Context-based access control (CBAC)
- Graph-based access control (GBAC)
- Lattice-based access control (LBAC)
- Organization-based access control (OrBAC)
- Rule-set-based access control (RSBAC)

Discretionary access control (DAC)

- Owner of a resource decides how it can be shared
- The owner can choose to give read, write, or other access to other users

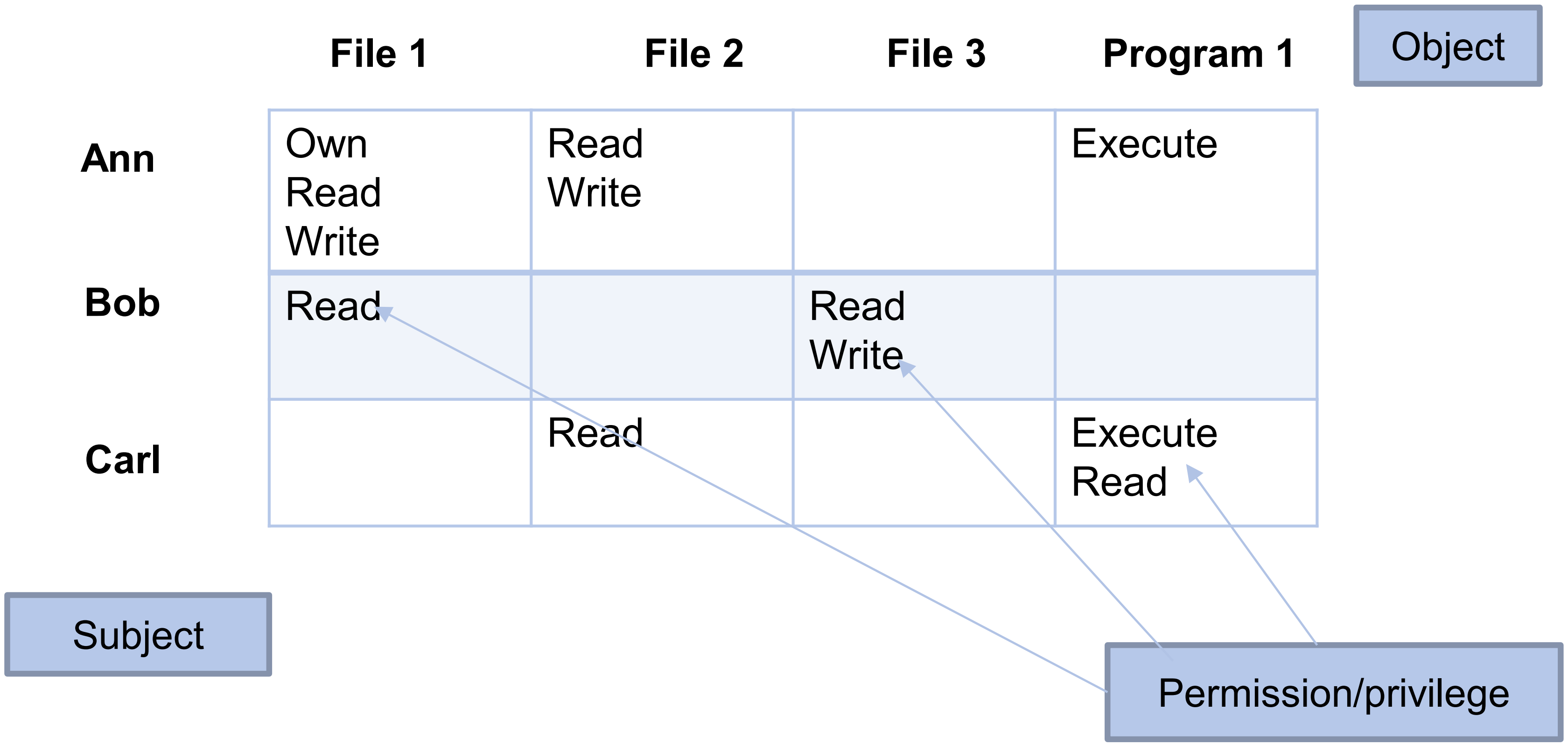


Object

Owner

Specifies users/groups who can access

Access control matrix



	File 1	File 2	File 3	Program 1
Ann	Own Read Write	Read Write		Execute
Bob	Read		Read Write	
Carl		Read		Execute Read

Object

Subject

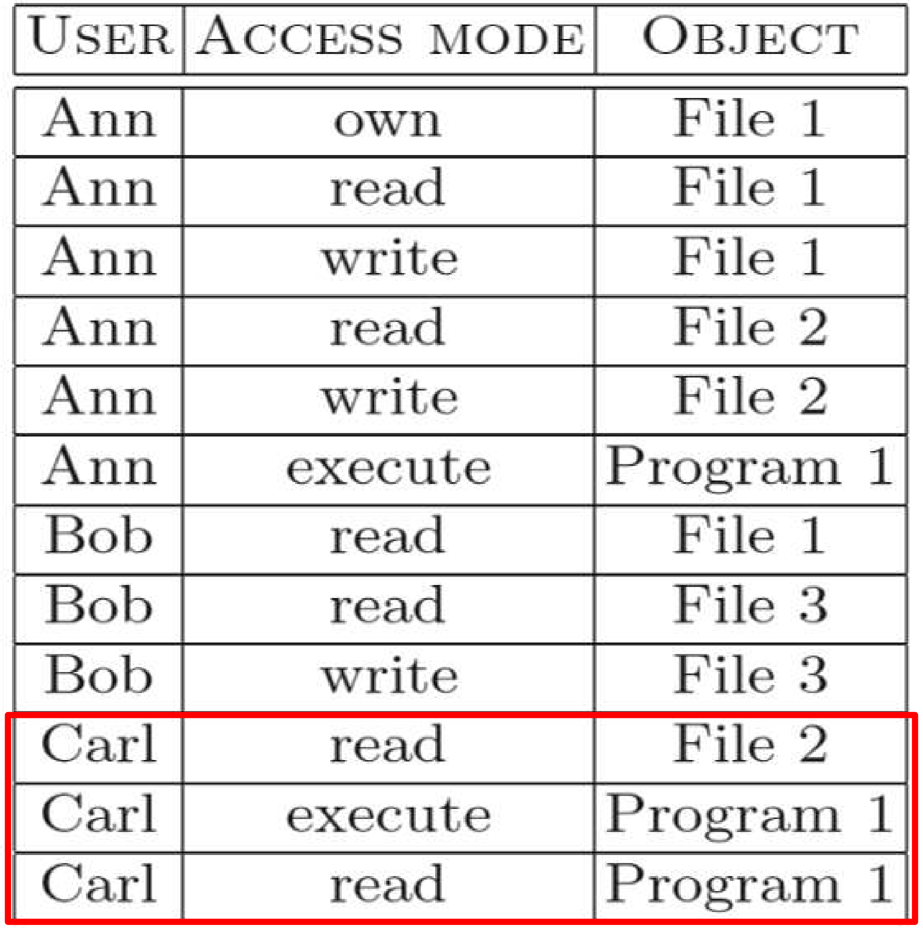
Permission/privilege

One mechanism to implement the matrix model

	File 1	File 2	File 3	Program 1
Ann	Own Read Write	Read Write		Execute
Bob	Read		Read Write	
Carl		Read		Execute Read

Just pick up non-empty entries and make a list, you get Authorization Table

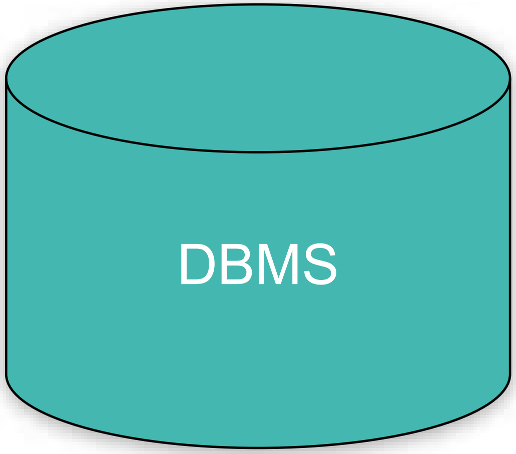
Authorization table

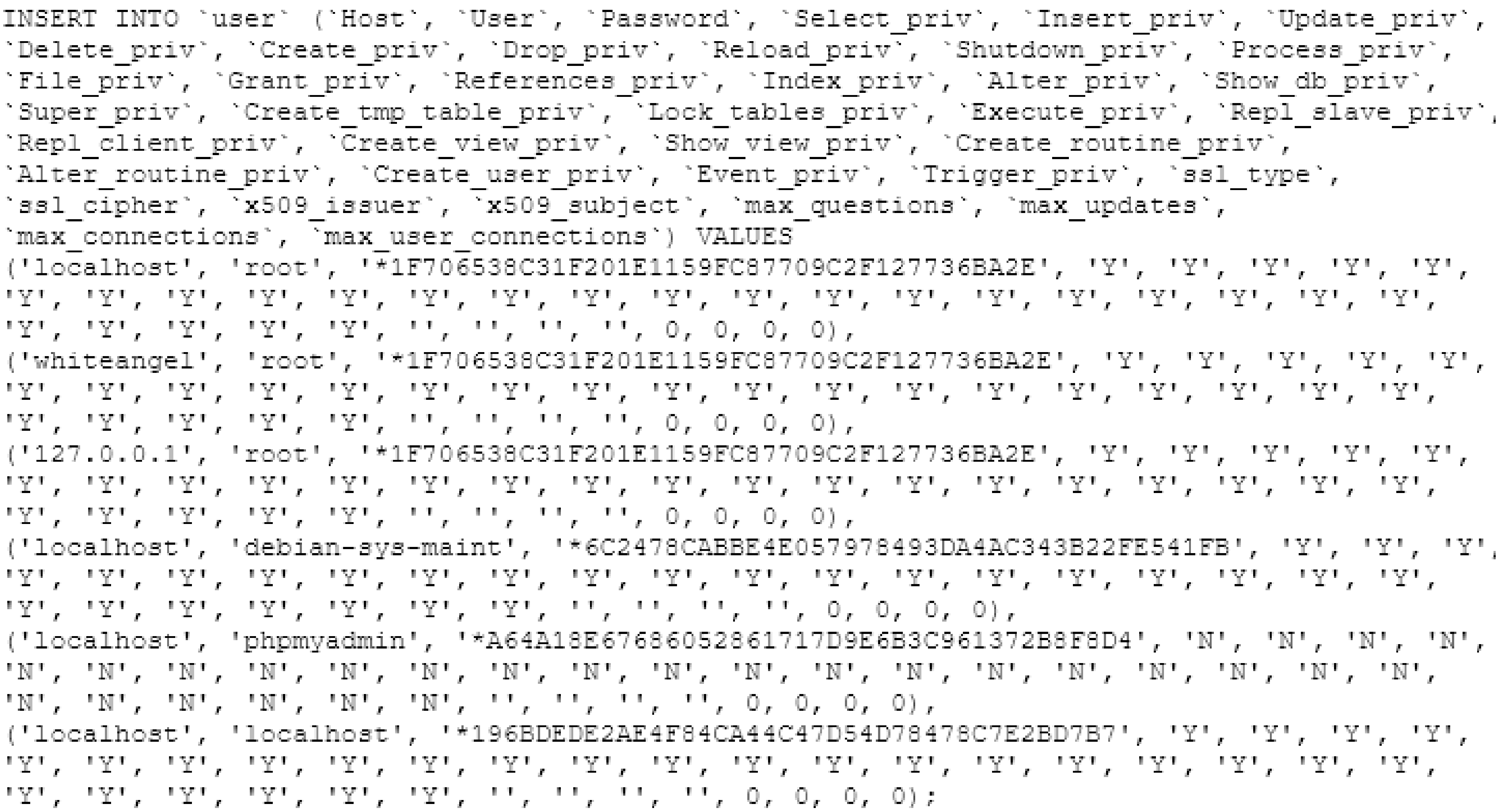


USER	ACCESS MODE	OBJECT
Ann	own	File 1
Ann	read	File 1
Ann	write	File 1
Ann	read	File 2
Ann	write	File 2
Ann	execute	Program 1
Bob	read	File 1
Bob	read	File 3
Bob	write	File 3
Carl	read	File 2
Carl	execute	Program 1
Carl	read	Program 1

Authorization table (cont')

- Generally used in DBMS
- Authorizations are stored as relational tables





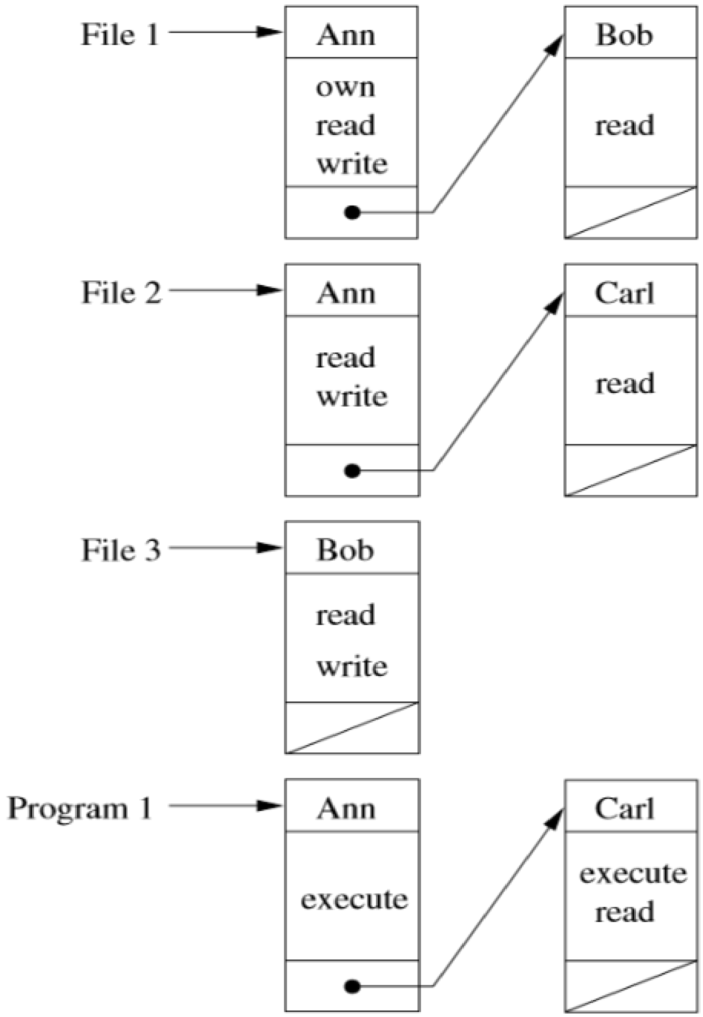
```
INSERT INTO `user` (`Host`, `User`, `Password`, `Select_priv`, `Insert_priv`, `Update_priv`,
`Delete_priv`, `Create_priv`, `Drop_priv`, `Reload_priv`, `Shutdown_priv`, `Process_priv`,
`File_priv`, `Grant_priv`, `References_priv`, `Index_priv`, `Alter_priv`, `Show_db_priv`,
`Super_priv`, `Create_tmp_table_priv`, `Lock_tables_priv`, `Execute_priv`, `Repl_slave_priv`,
`Repl_client_priv`, `Create_view_priv`, `Show_view_priv`, `Create_routine_priv`,
`Alter_routine_priv`, `Create_user_priv`, `Event_priv`, `Trigger_priv`, `ssl_type`,
`ssl_cipher`, `x509_issuer`, `x509_subject`, `max_questions`, `max_updates`,
`max_connections`, `max_user_connections`) VALUES
('localhost', 'root', '*1F706538C31F201E1159FC87709C2F127736BA2E', 'Y', 'Y', 'Y', 'Y', 'Y',
'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
'Y', 'Y', 'Y', 'Y', 'Y', '', '', '', '', 0, 0, 0, 0),
('whiteangel', 'root', '*1F706538C31F201E1159FC87709C2F127736BA2E', 'Y', 'Y', 'Y', 'Y', 'Y',
'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
'Y', 'Y', 'Y', 'Y', 'Y', '', '', '', '', 0, 0, 0, 0),
('127.0.0.1', 'root', '*1F706538C31F201E1159FC87709C2F127736BA2E', 'Y', 'Y', 'Y', 'Y', 'Y',
'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
'Y', 'Y', 'Y', 'Y', 'Y', '', '', '', '', 0, 0, 0, 0),
('localhost', 'debian-sys-maint', '*6C2478CABBE4E057978493DA4AC343B22FE541FB', 'Y', 'Y', 'Y',
'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', '', '', '', '', 0, 0, 0, 0),
('localhost', 'phpmyadmin', '*A64A18E67686052861717D9E6B3C961372B8F8D4', 'N', 'N', 'N', 'N',
'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N',
'N', 'N', 'N', 'N', 'N', 'N', '', '', '', '', 0, 0, 0, 0),
('localhost', 'localhost', '*196BDEDE2AE4F84CA44C47D54D78478C7E2BD7B7', 'Y', 'Y', 'Y', 'Y',
'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
'Y', 'Y', 'Y', 'Y', 'Y', 'Y', '', '', '', '', 0, 0, 0, 0);
```

Another mechanism to implement the matrix model

	File 1	File 2	File 3	Program 1
Ann	Own Read Write	Read Write		Execute
Bob	Read		Read Write	
Carl		Read		Execute Read

Store information according to objects, you get Access control list (ACL)

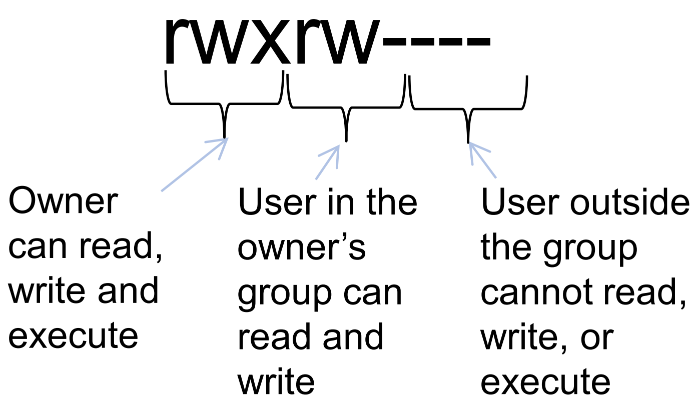
Access control list (ACL)



Access control list (ACL) (Cont')

- Used in modern OS

Linux command: ls –l

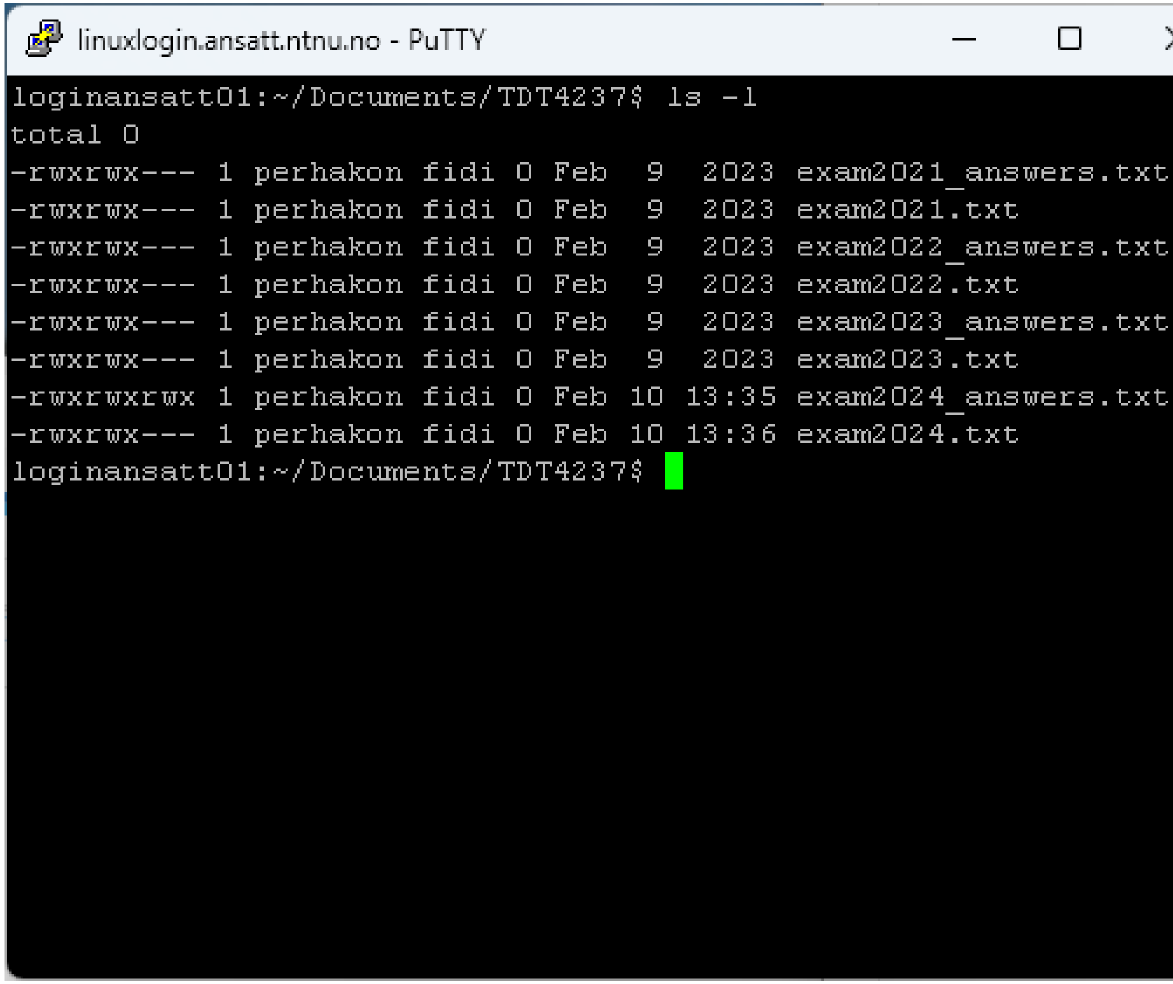


rwxrw----

Owner can read, write and execute

User in the owner's group can read and write

User outside the group cannot read, write, or execute



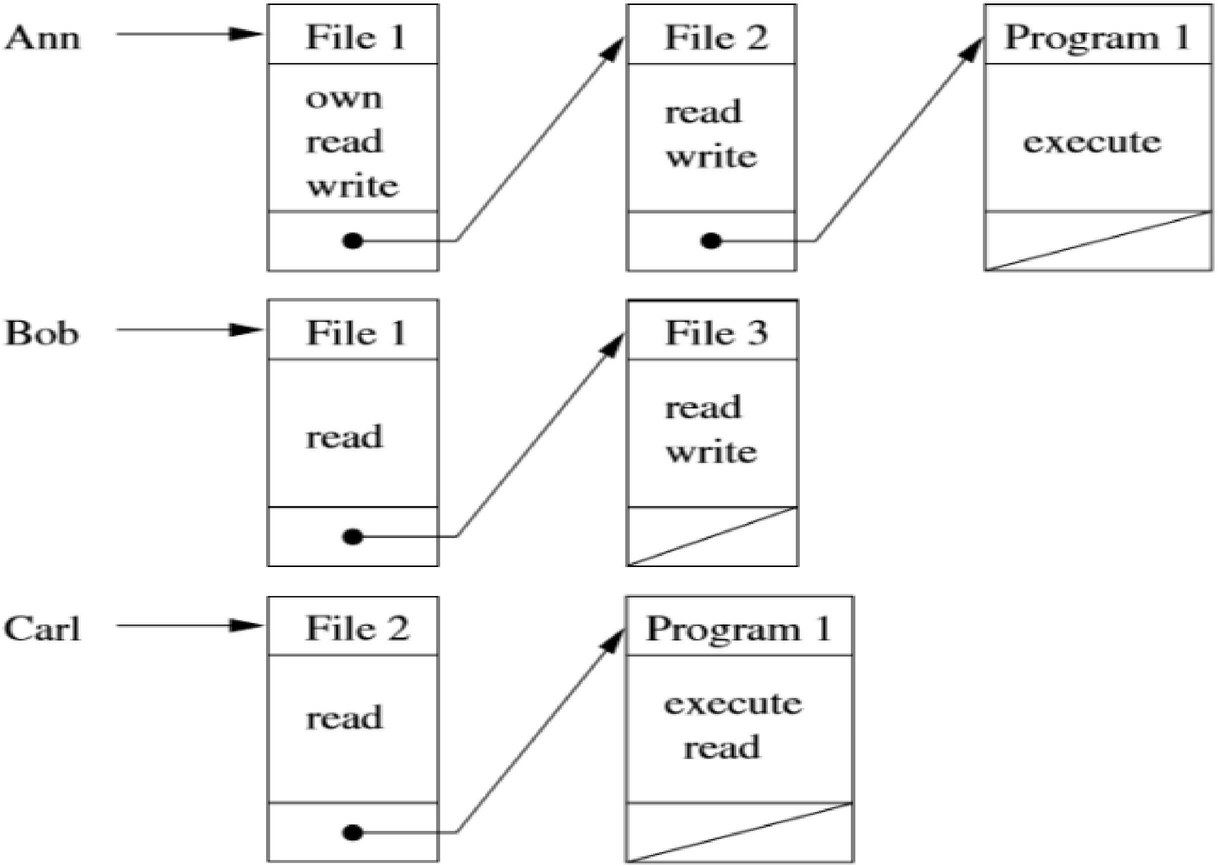
```
loginansatt01:~/Documents/TDT4237$ ls -l
total 0
-rwxrwx--- 1 perhakon fidi 0 Feb  9  2023 exam2021_answers.txt
-rwxrwx--- 1 perhakon fidi 0 Feb  9  2023 exam2021.txt
-rwxrwx--- 1 perhakon fidi 0 Feb  9  2023 exam2022_answers.txt
-rwxrwx--- 1 perhakon fidi 0 Feb  9  2023 exam2022.txt
-rwxrwx--- 1 perhakon fidi 0 Feb  9  2023 exam2023_answers.txt
-rwxrwx--- 1 perhakon fidi 0 Feb  9  2023 exam2023.txt
-rwxrwxrwx 1 perhakon fidi 0 Feb 10 13:35 exam2024_answers.txt
-rwxrwx--- 1 perhakon fidi 0 Feb 10 13:36 exam2024.txt
loginansatt01:~/Documents/TDT4237$ 
```

The third mechanism to implement the matrix model

	File 1	File 2	File 3	Program 1
Ann	Own Read Write	Read Write		Execute
Bob	Read		Read Write	
Carl		Read		Execute Read

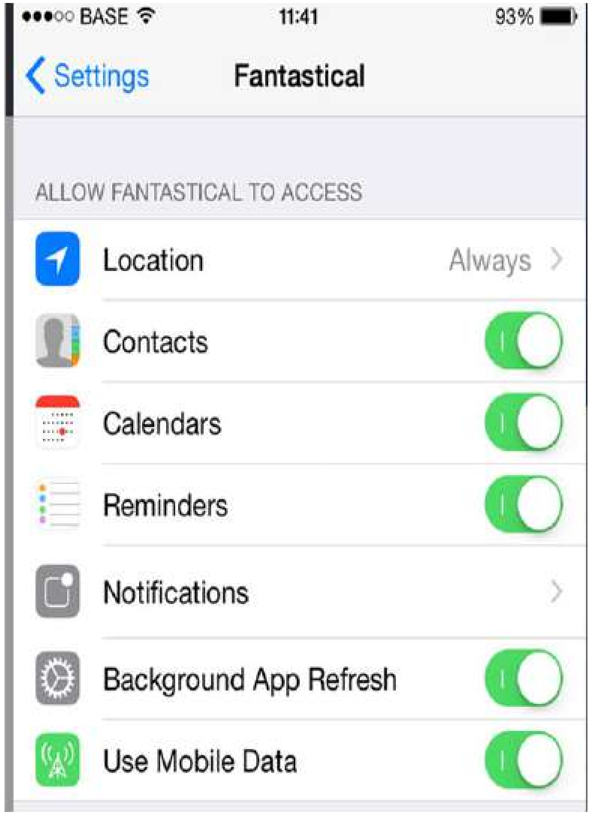
Store information according to the subject, you get Capability

Capabilities



Capability (Cont’)

- iOS permission control
- Data is segregated into classes, e.g.
  - Contacts, calendar, photos, reminders, etc.
- Only allow very basic permission at installation
- At runtime, app must ask user to get more permissions



●●●○○ BASE 11:41 93%

< Settings Fantastical

ALLOW FANTASTICAL TO ACCESS

Location: Always >

Contacts: On

Calendars: On

Reminders: On

Notifications >

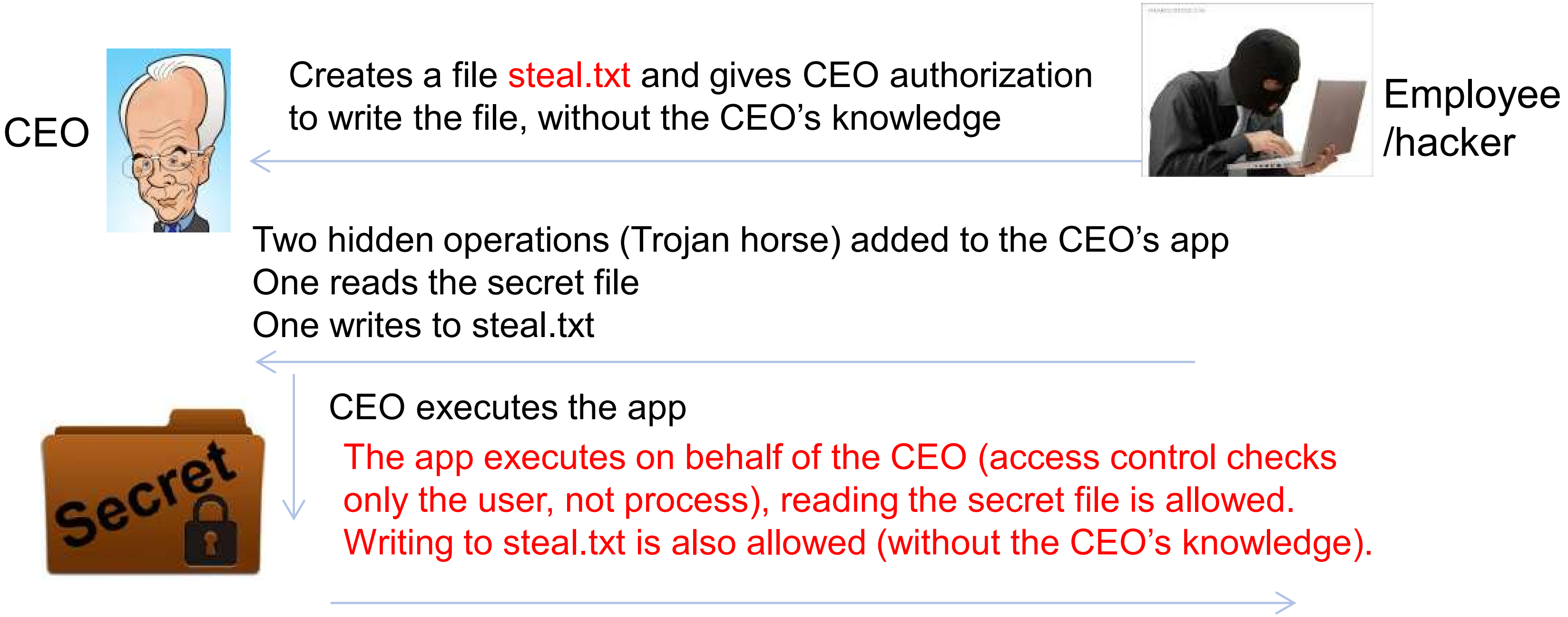
Background App Refresh: On

Use Mobile Data: On

Vulnerabilities of DAC

- Does not distinguish between *user* and *process*

Vulnerable to a process executing malicious programs (Trojan Horse) exploiting the authorization of the user



CEO

Creates a file steal.txt and gives CEO authorization to write the file, without the CEO's knowledge

Employee /hacker

Two hidden operations (Trojan horse) added to the CEO's app
One reads the secret file
One writes to steal.txt

CEO executes the app

The app executes on behalf of the CEO (access control checks only the user, not process), reading the secret file is allowed. Writing to steal.txt is also allowed (without the CEO's knowledge).

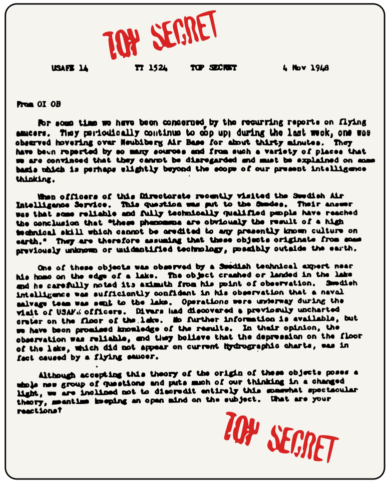
Mandatory Access Control

- Unlike discretionary access control (DAC) where users can take their own access decisions about their files

- Mandatory access control (MAC) means that systems enforce a security policy independent of the user’s action

Mandatory Access Control (Cont')

- Enforce access control on the basis of regulations mandated by a central authority
- Access class is assigned to each object and subject



TOP SECRET

USAFE 14 TT 1524 TOP SECRET 4 Nov 1948

From OI OB

For some time we have been concerned by the recurring reports on flying saucers. They periodically continue to pop up; during the last week, one was observed hovering over Neubiberg Air Base for about thirty minutes. They have been reported by so many sources and from such a variety of places that we are convinced that they cannot be disregarded and must be explained on some basis which is perhaps slightly beyond the scope of our present intelligence thinking.

When officers of this Directorate recently visited the Swedish Air Intelligence Service. This question was put to the Swedes. Their answer was that some reliable and fully technically qualified people have reached the conclusion that "these phenomena are obviously the result of a high technical skill which cannot be credited to any presently known culture on earth." They are therefore assuming that these objects originate from some previously unknown or unidentified technology, possibly outside the earth.

One of these objects was observed by a Swedish technical expert near his home on the edge of a lake. The object crashed or landed in the lake and he carefully noted its azimuth from his point of observation. Swedish intelligence was sufficiently confident in his observation that a naval salvage team was sent to the lake. Operations were underway during the visit of USAFE officers. Divers had discovered a previously uncharted crater on the floor of the lake. No further information is available, but we have been promised knowledge of the results. In their opinion, the observation was reliable, and they believe that the depression on the floor of the lake, which did not appear on current Hydrographic charts, was in fact caused by a flying saucer.

Although accepting this theory of the origin of these objects poses a whole new group of questions and puts much of our thinking in a changed light, we are inclined not to discredit entirely this somewhat spectacular theory, meantime keeping an open mind on the subject. What are your reactions?

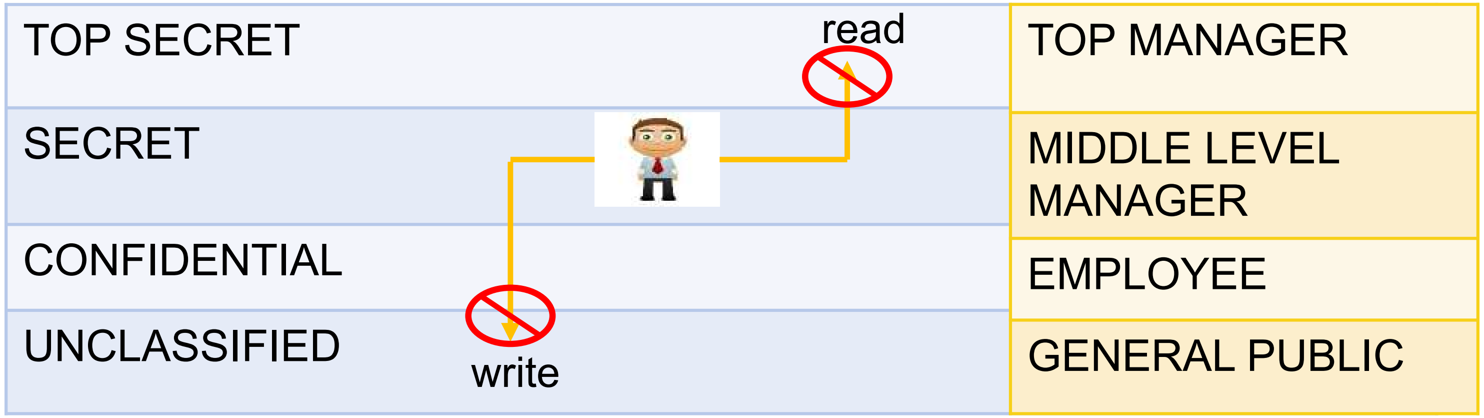
TOP SECRET

Object classification
TOP SECRET
SECRET
CONFIDENTIAL
UNCLASSIFIED

Subject classification
TOP MANAGER
MIDDLE LEVEL MANAGER
EMPLOYEE
GENERAL PUBLIC

Bell-LaPadula model

No read up (NRU)



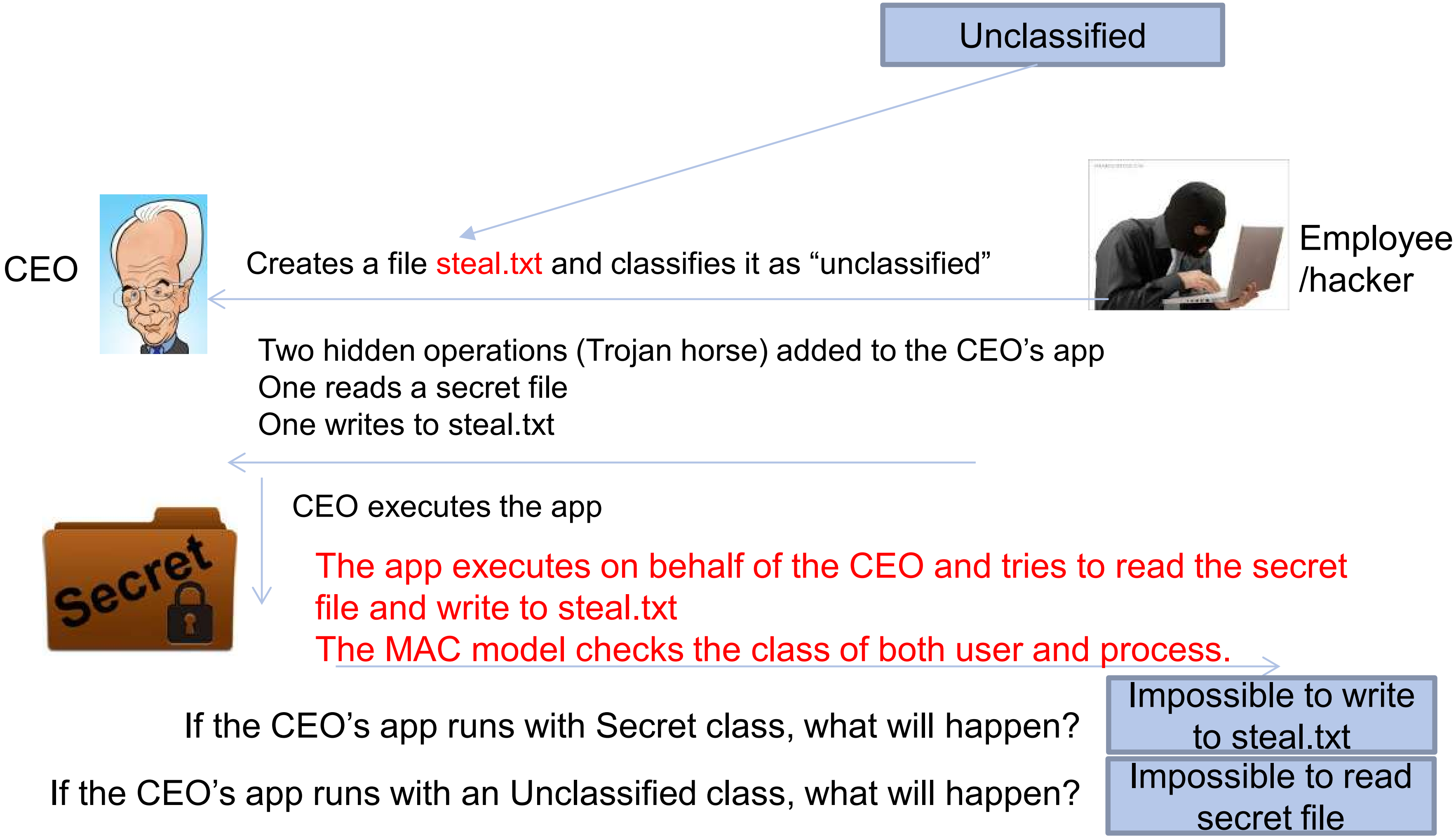
TOP SECRET	TOP MANAGER
SECRET	MIDDLE LEVEL MANAGER
CONFIDENTIAL	EMPLOYEE
UNCLASSIFIED	GENERAL PUBLIC

No write down (NWD)
(* property)

Confidentiality

Strong *: Only operations on the same level

Why Bell-LaPadula model works?



Unclassified

CEO

Creates a file steal.txt and classifies it as “unclassified”

Employee /hacker

Two hidden operations (Trojan horse) added to the CEO’s app
One reads a secret file
One writes to steal.txt

CEO executes the app

The app executes on behalf of the CEO and tries to read the secret file and write to steal.txt
The MAC model checks the class of both user and process.

If the CEO’s app runs with Secret class, what will happen?

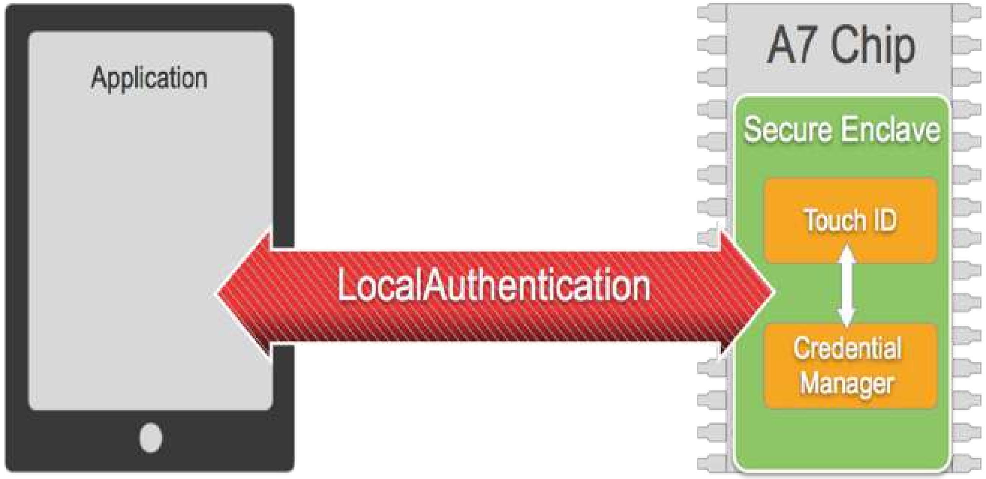
Impossible to write to steal.txt

If the CEO’s app runs with an Unclassified class, what will happen?

Impossible to read secret file

An example application of Bell-LaPadula model

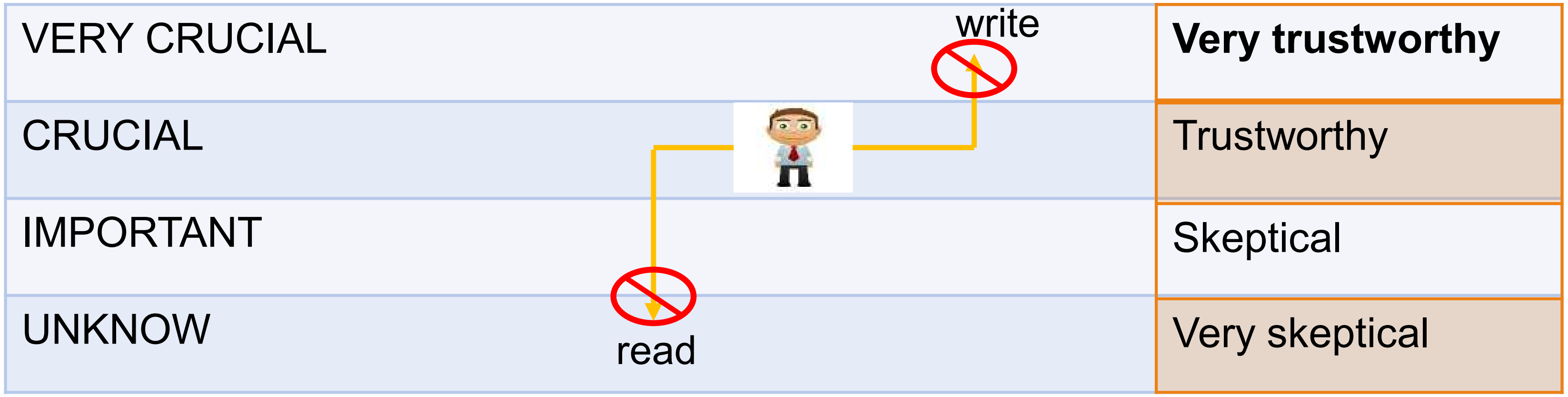
- No read up
- No write down



iOS: secure enclave: preventing applications from reading the security keys

Biba model

No write up (NWU)



VERY CRUCIAL	**Very trustworthy**
CRUCIAL	Trustworthy
IMPORTANT	Skeptical
UNKNOW	Very skeptical

No read down (NRD)

Integrity

Why Biba model works?

- No improper modification of high integrity objects from the low classified subject (No write up), e.g.,
  - Software downloaded from the web cannot write to OS
- High integrity object is not contaminated due to reading and using unreliable data (No read down), e.g.,
  - Signaling sys. does not use data from passenger info. sys.

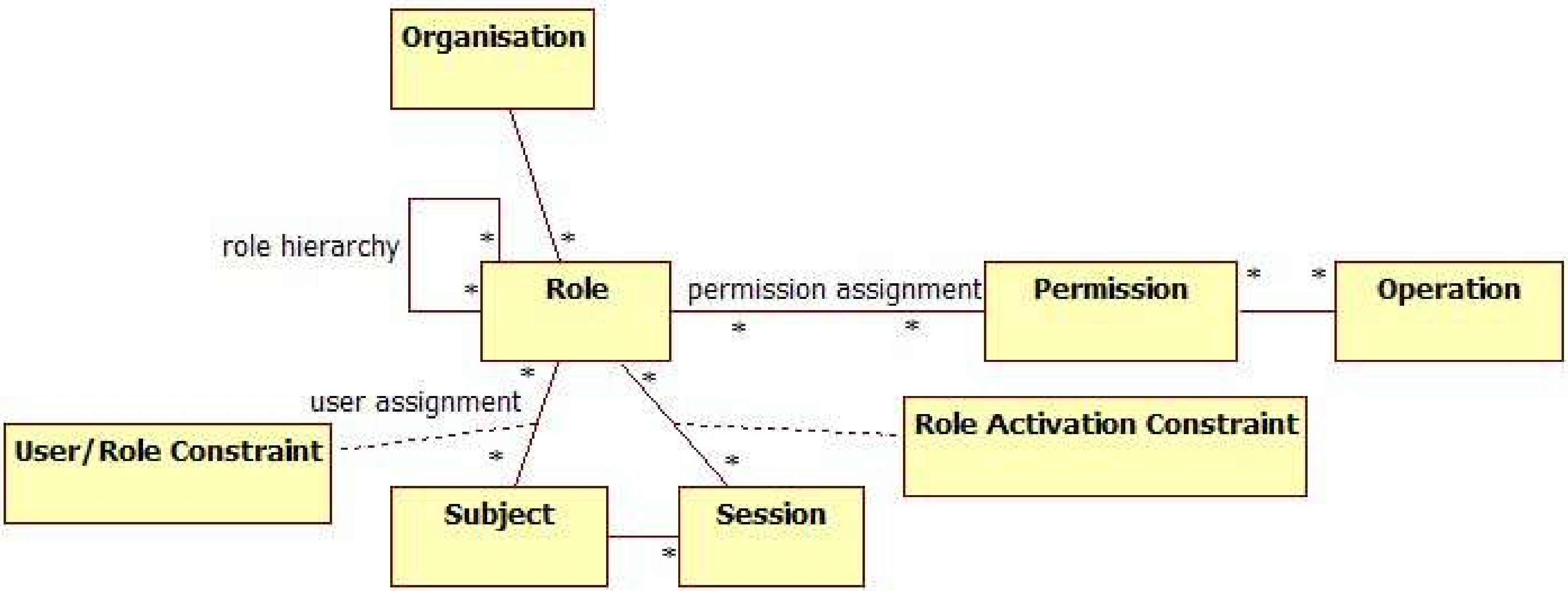
Combine Bell-LaPadula and Biba model

- If both ***confidentiality*** and ***integrity*** have to be controlled
- Objects and subjects have to be assigned to two access classes
  - One for confidentiality control
  - One for integrity control

DAC vs. MAC

DAC	MAC
Advantages: • Simple and efficient access right management • Scalability	Advantage: • Strict control over information flow • Strong exploit containment
Disadvantages: • Weak control over information flow	Disadvantages: • Cumbersome administration

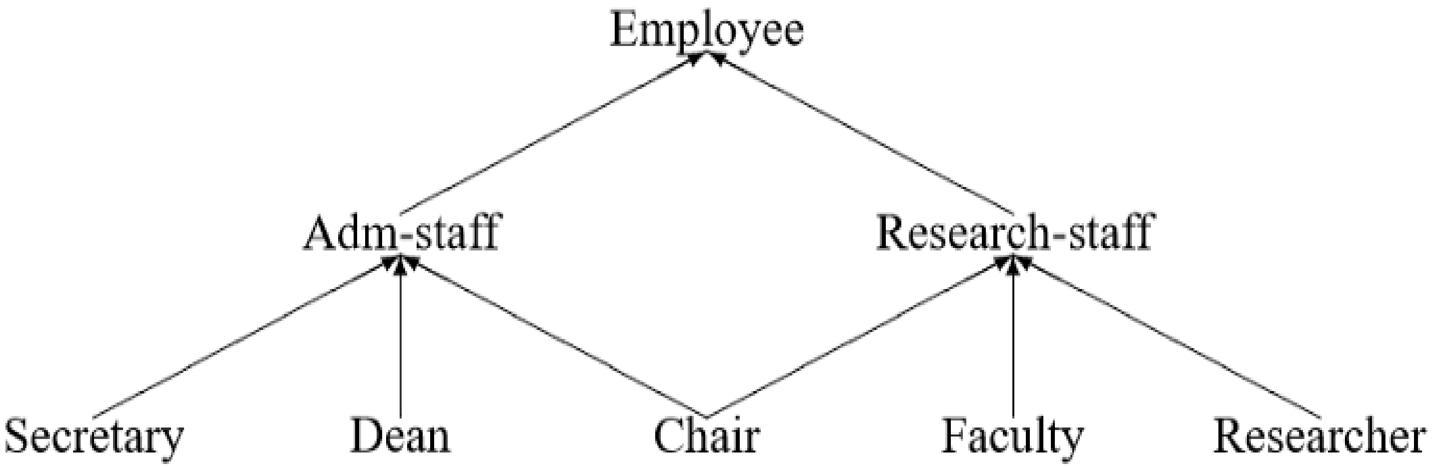
Role-Based Access Control



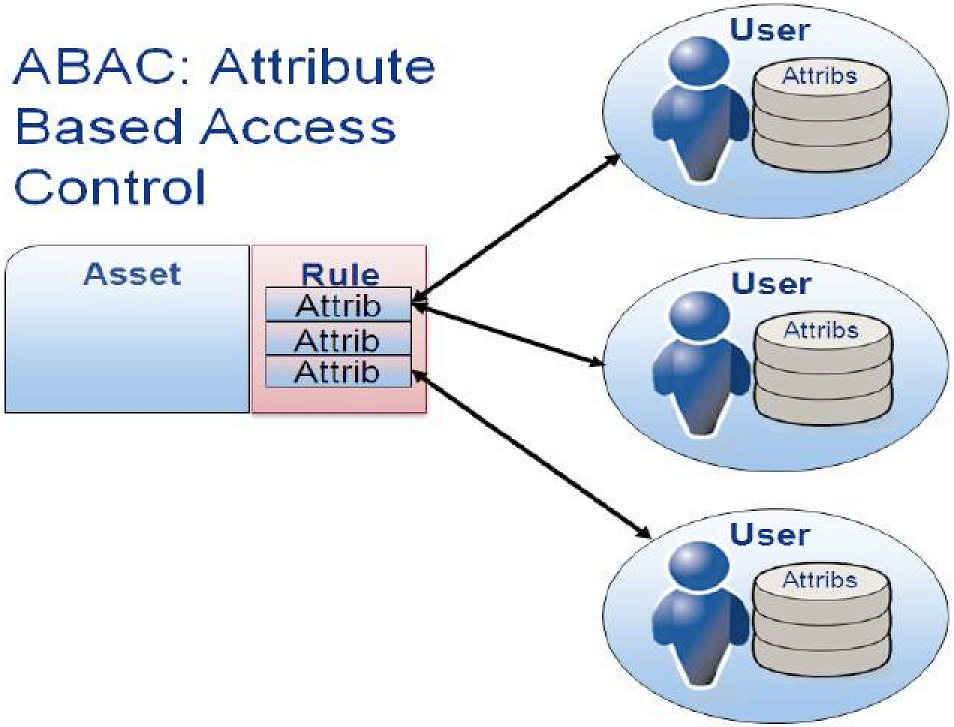
https://en.wikipedia.org/wiki/Role-based_access_control#/media/File:Role-based_access_control.jpg

Benefits of RBAC

- Easy authorization management
- Maps to real-world role hierarchy



Attribute-Based Access Control



≈Aspect-based access control

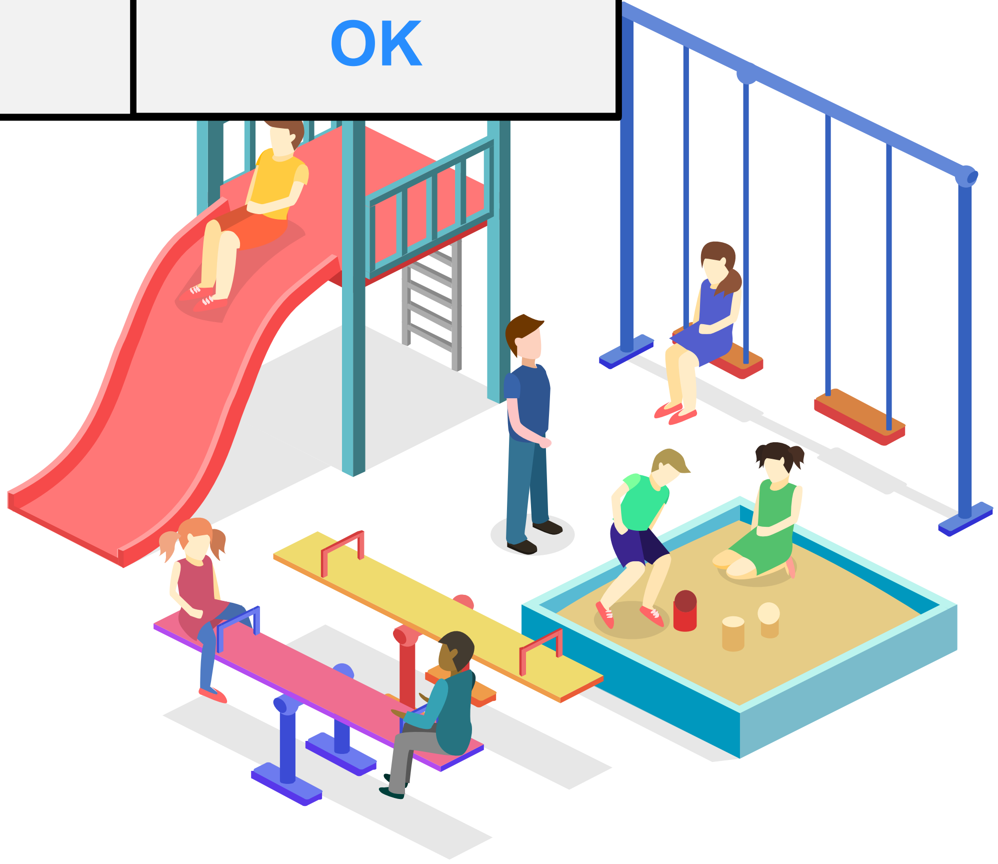
- **RBAC is for coarse-grain access control**
- **ABAC is for fine-grain access controls (more difficult to use correctly)**
- **RBAC before ABAC (who can see what module BEFORE what can they see inside a module)**

Browsers

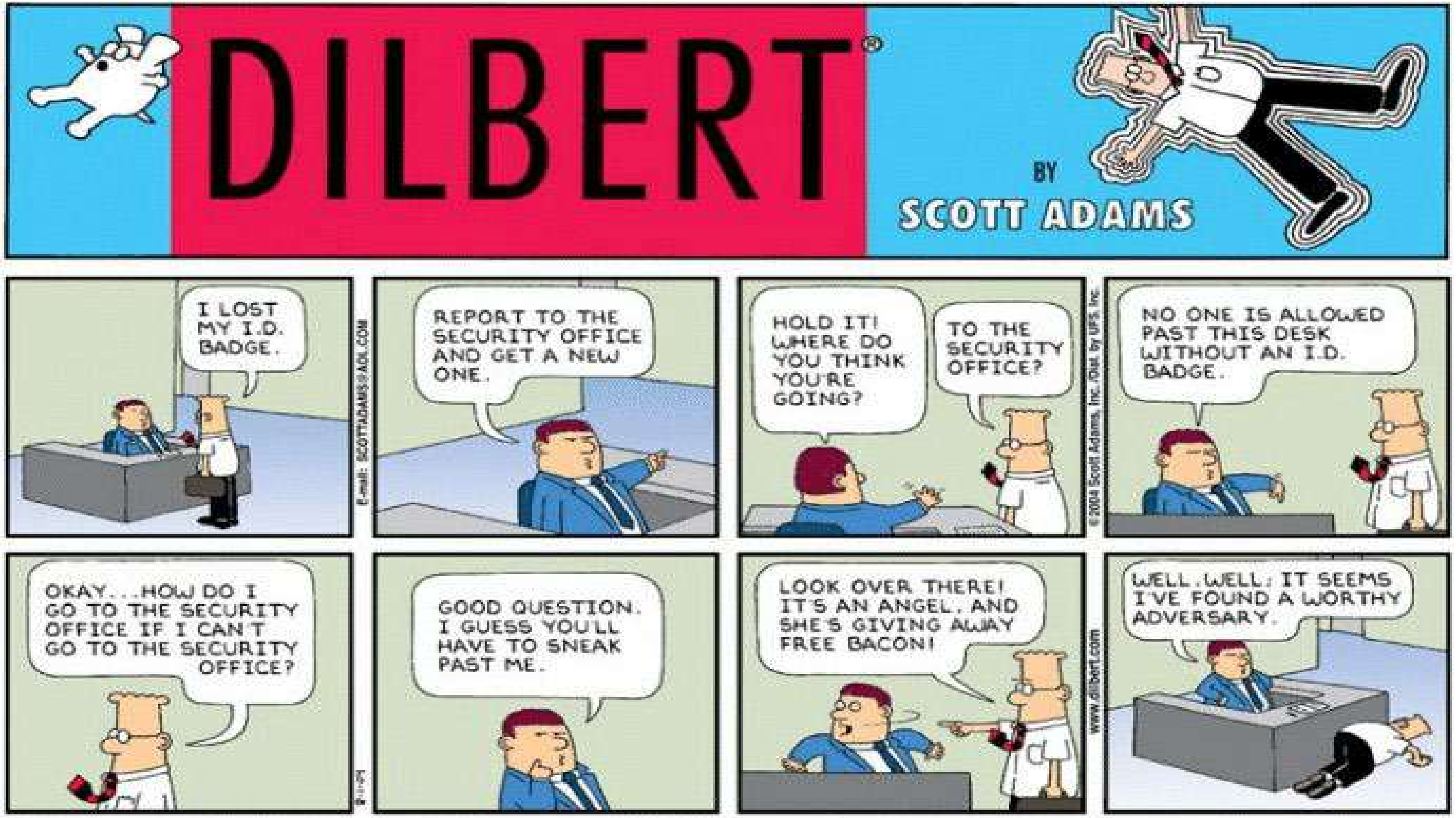
Allow evil.com to access your soul?

Don’t allow | **OK**

- **Same-origin policy:** Only communicate with the IP you originate from
- **Sandbox:** Restricted environment
- Ask user for more…



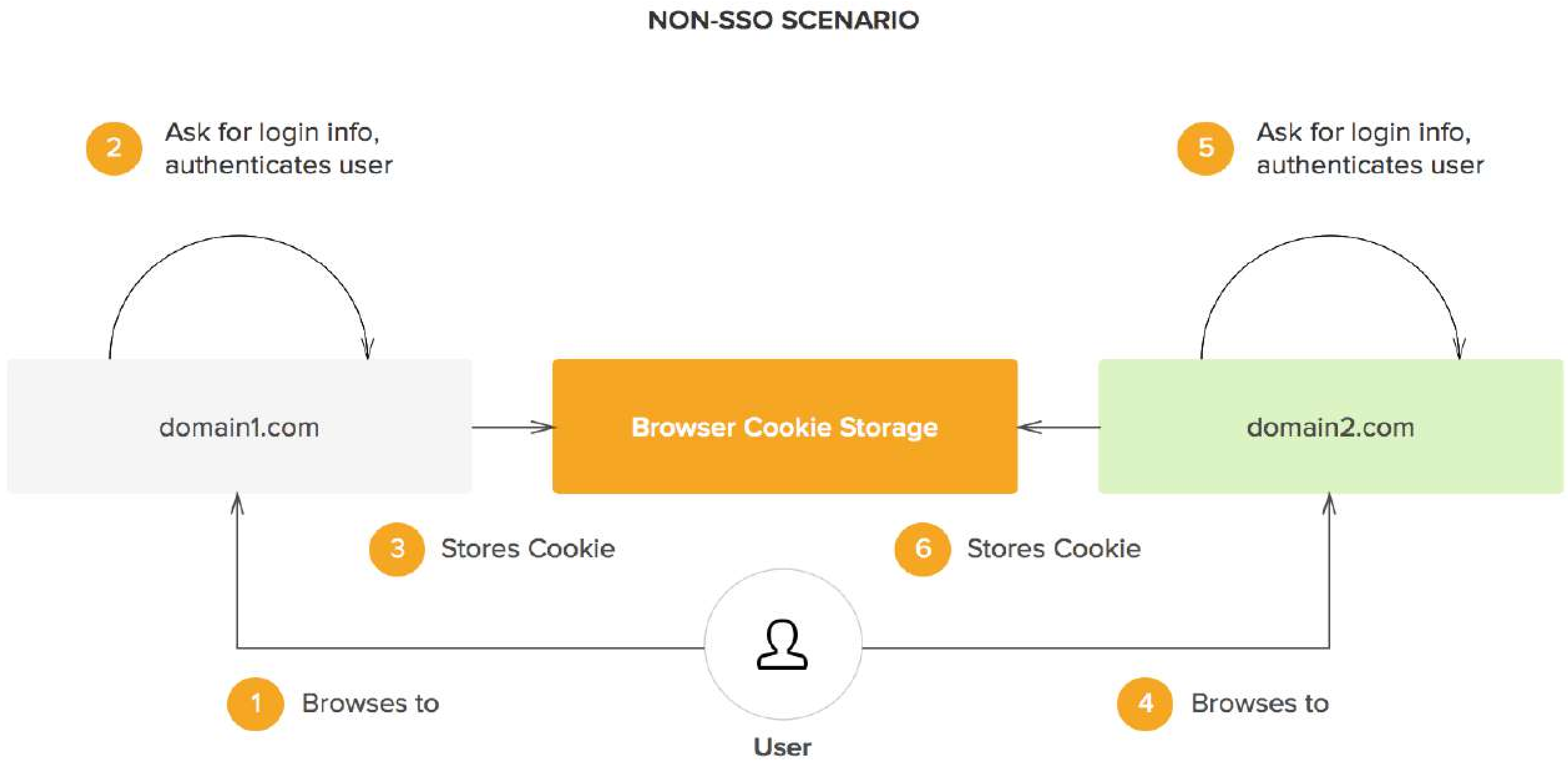
Access control operation


DILBERT®
BY SCOTT ADAMS
I LOST MY I.D. BADGE.
E-mail: SCOTTADAMS@AOL.COM
REPORT TO THE SECURITY OFFICE AND GET A NEW ONE.
HOLD IT! WHERE DO YOU THINK YOU'RE GOING?
TO THE SECURITY OFFICE?
© 2004 Scott Adams, Inc./Dist. by UFS Inc.
NO ONE IS ALLOWED PAST THIS DESK WITHOUT AN I.D. BADGE.
OKAY... HOW DO I GO TO THE SECURITY OFFICE IF I CAN'T GO TO THE SECURITY OFFICE?
GOOD QUESTION. I GUESS YOU'LL HAVE TO SNEAK PAST ME.
LOOK OVER THERE! IT'S AN ANGEL, AND SHE'S GIVING AWAY FREE BACON!
www.dilbert.com
WELL, WELL: IT SEEMS I'VE FOUND A WORTHY ADVERSARY.

© UFS, Inc.

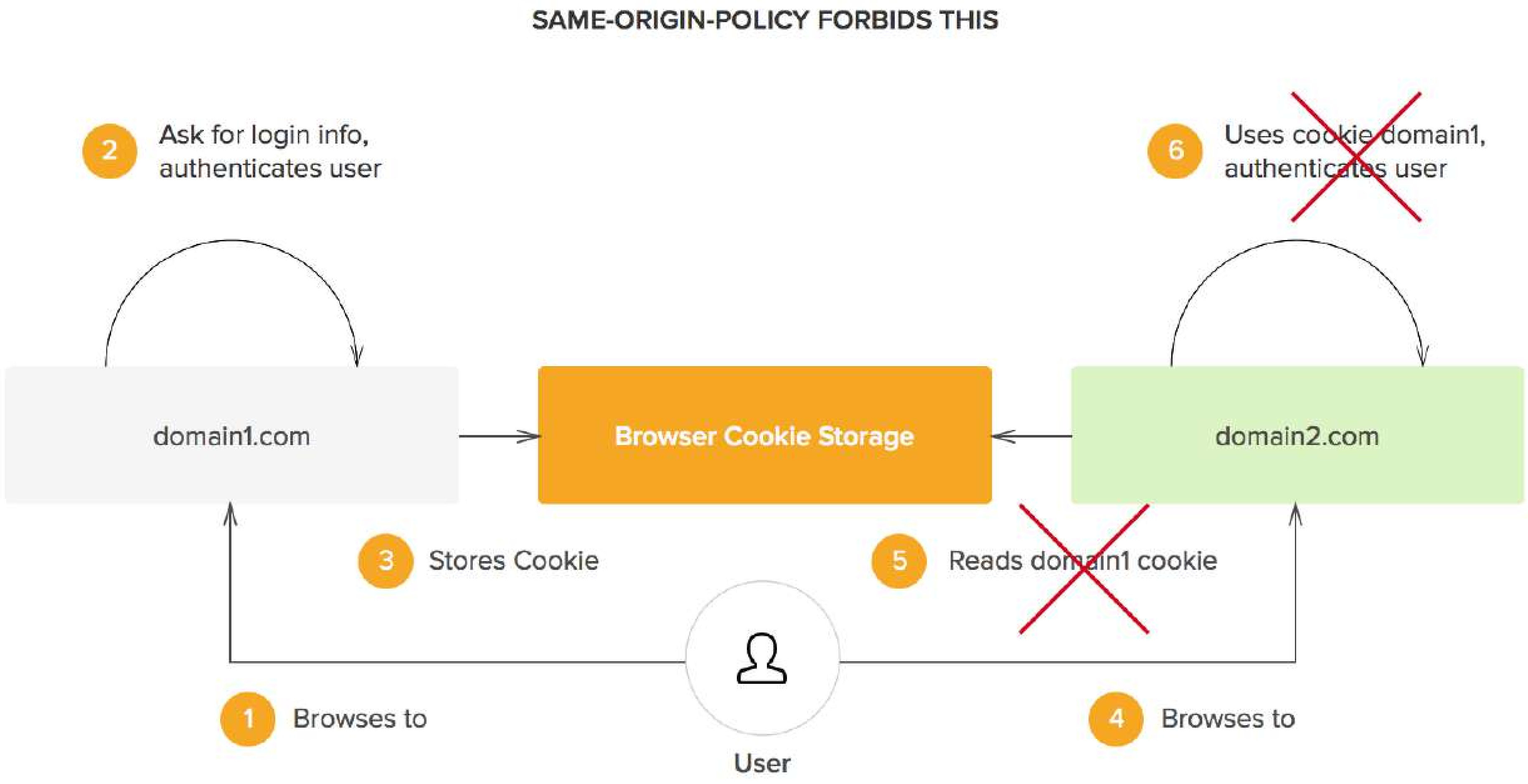
Authentication and SSO

Without Single Sign-On (SSO)*



*https://auth0.com/blog/what-is-and-how-does-single-sign-on-work/

Without Single Sign-On (SSO) (Cont')*

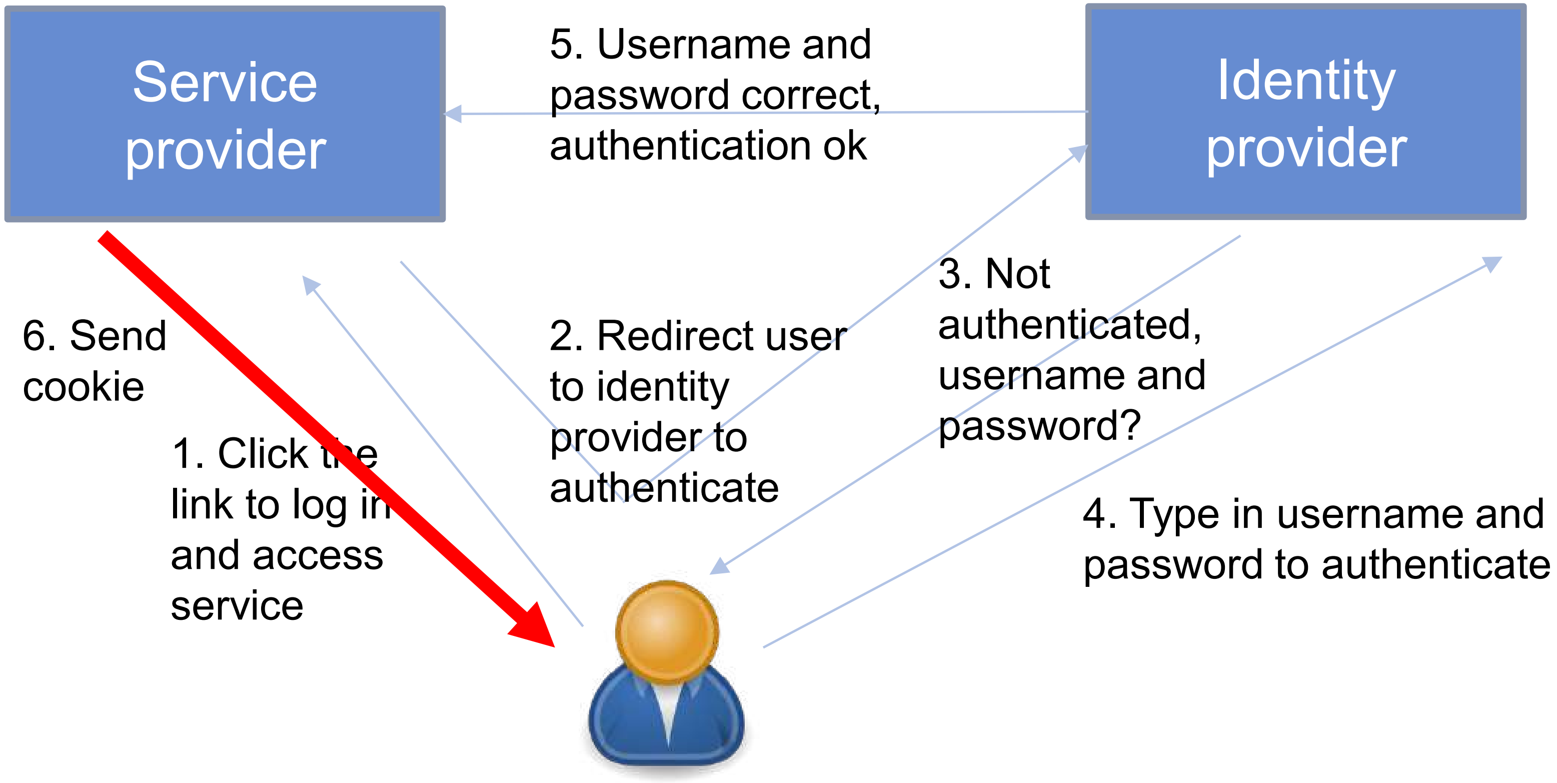

SAME-ORIGIN-POLICY FORBIDS THIS
2 Ask for login info, authenticates user
6 Uses cookie domain1, authenticates user
domain1.com
Browser Cookie Storage
domain2.com
3 Stores Cookie
5 Reads domain1 cookie
1 Browses to
4 Browses to
User

*https://auth0.com/blog/what-is-and-how-does-single-sign-on-work/

Challenges of Non-SSO

- User
  - Not user-friendly
- Administrator/developer
  - Hard to manage authentication of multiple apps
  - Security risks

Single Sign-On


Service provider

Identity provider

5. Username and password correct, authentication ok

6. Send cookie

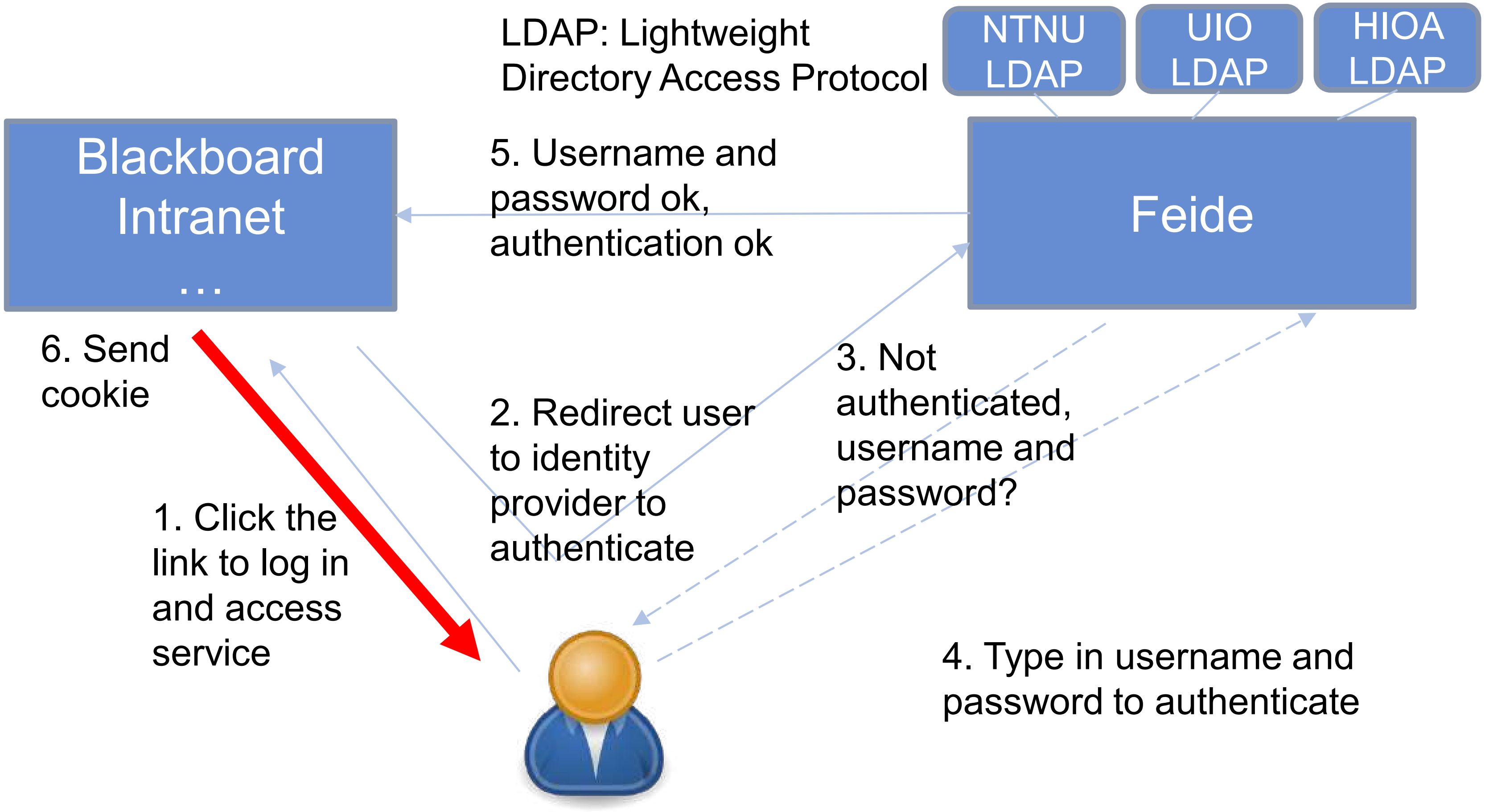
1. Click the link to log in and access service

2. Redirect user to identity provider to authenticate

3. Not authenticated, username and password?

4. Type in username and password to authenticate

Single Sign-On at NTNU



LDAP: Lightweight Directory Access Protocol

1. Click the link to log in and access service

2. Redirect user to identity provider to authenticate

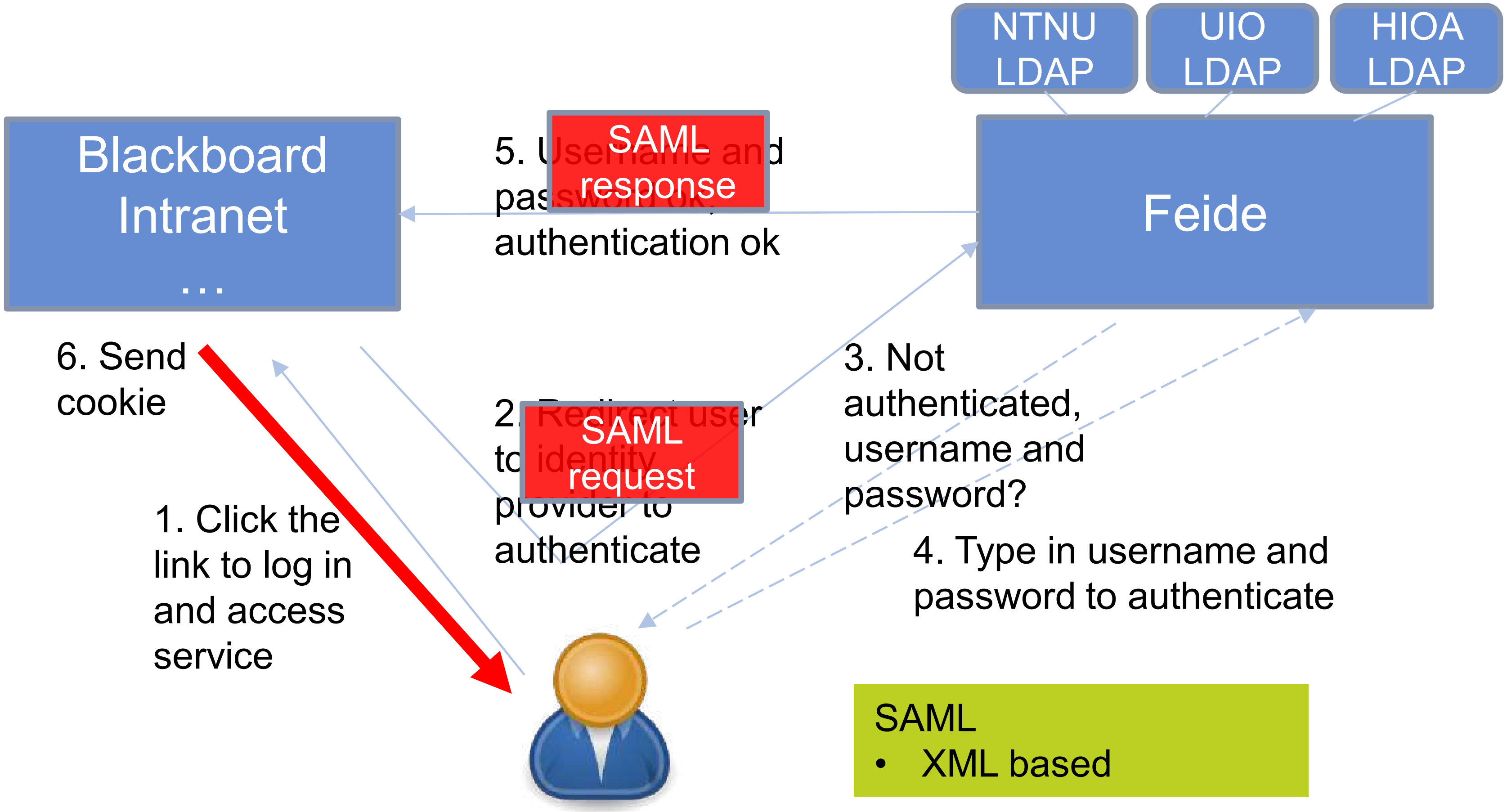
3. Not authenticated, username and password?

4. Type in username and password to authenticate

5. Username and password ok, authentication ok

6. Send cookie

Feide uses SAML (Security Assertion Markup Language) 2.0



NTNU LDAP

UIO LDAP

HIOA LDAP

Blackboard Intranet …

Feide

5. Username and password ok, authentication ok

SAML response

6. Send cookie

1. Click the link to log in and access service

2. Redirect user to identity provider to authenticate

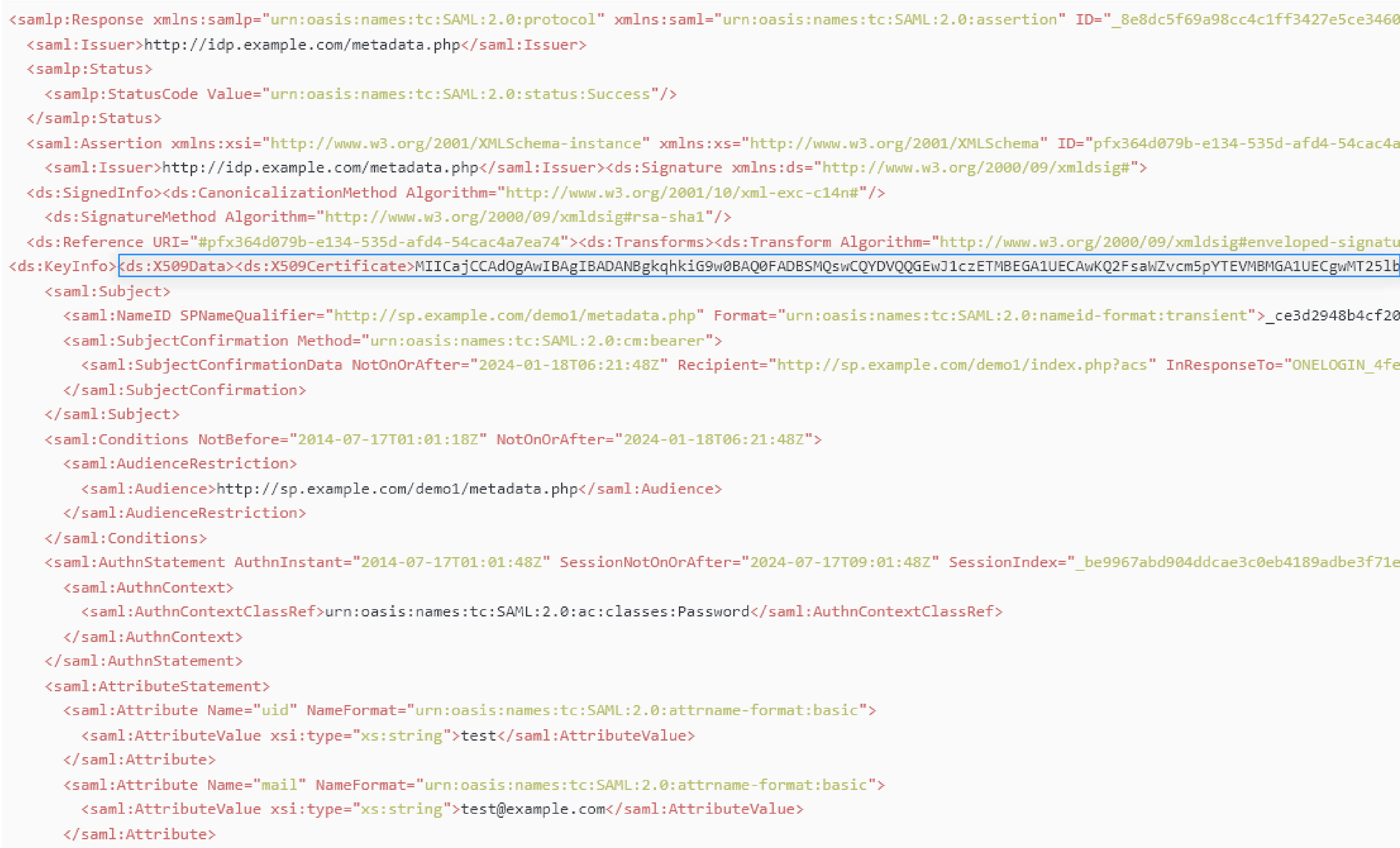
SAML request

3. Not authenticated, username and password?

4. Type in username and password to authenticate

SAML
- XML based

See more on: https://docs.feide.no/reference/saml/saml2_technical_guide.html

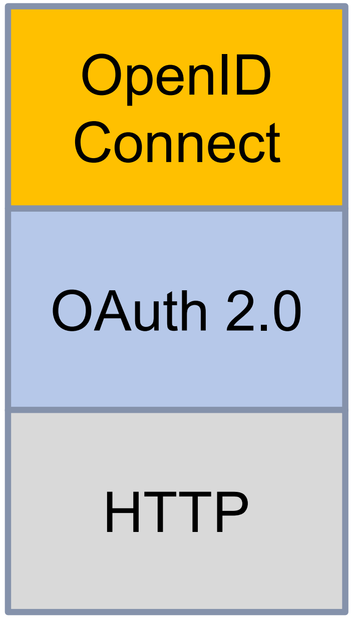


```xml
<samlp:Response xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol" xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" ID="_8e8dc5f69a98cc4c1ff3427e5ce3460
  <saml:Issuer>http://idp.example.com/metadata.php</saml:Issuer>
  <samlp:Status>
    <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
  </samlp:Status>
  <saml:Assertion xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs="http://www.w3.org/2001/XMLSchema" ID="pfx364d079b-e134-535d-afd4-54cac4a
    <saml:Issuer>http://idp.example.com/metadata.php</saml:Issuer><ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <ds:SignedInfo><ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
    <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
  <ds:Reference URI="#pfx364d079b-e134-535d-afd4-54cac4a7ea74"><ds:Transforms><ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signatu
<ds:KeyInfo><ds:X509Data><ds:X509Certificate>MIICajCCAdOgAwIBAgIBADANBgkqhkiG9w0BAQ0FADBSMQswCQYDVQQGEwJ1czETMBEGA1UECAwKQ2FsaWZvcm5pYTEVMBMGA1UECgwMT25lb
    <saml:Subject>
      <saml:NameID SPNameQualifier="http://sp.example.com/demo1/metadata.php" Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient">_ce3d2948b4cf20
      <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
        <saml:SubjectConfirmationData NotOnOrAfter="2024-01-18T06:21:48Z" Recipient="http://sp.example.com/demo1/index.php?acs" InResponseTo="ONELOGIN_4fe
      </saml:SubjectConfirmation>
    </saml:Subject>
    <saml:Conditions NotBefore="2014-07-17T01:01:18Z" NotOnOrAfter="2024-01-18T06:21:48Z">
      <saml:AudienceRestriction>
        <saml:Audience>http://sp.example.com/demo1/metadata.php</saml:Audience>
      </saml:AudienceRestriction>
    </saml:Conditions>
    <saml:AuthnStatement AuthnInstant="2014-07-17T01:01:48Z" SessionNotOnOrAfter="2024-07-17T09:01:48Z" SessionIndex="_be9967abd904ddcae3c0eb4189adbe3f71e
      <saml:AuthnContext>
        <saml:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:Password</saml:AuthnContextClassRef>
      </saml:AuthnContext>
    </saml:AuthnStatement>
    <saml:AttributeStatement>
      <saml:Attribute Name="uid" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
        <saml:AttributeValue xsi:type="xs:string">test</saml:AttributeValue>
      </saml:Attribute>
      <saml:Attribute Name="mail" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
        <saml:AttributeValue xsi:type="xs:string">test@example.com</saml:AttributeValue>
      </saml:Attribute>
```

SSO Trends

- From SOAP/XML to more lightweight HTTP/JSON
- Social Sign-in (Facebook, Google, etc.)
- OpenID Connect (Authentication) and OAuth 2.0 (Authorization)
- From authentication only to API authorization (and data access)

OpenID Connect



OpenID Connect

OAuth 2.0

HTTP

OpenID Connect is for Authentication (Use ID Token)

OAuth 2.0 is for Authorization (Use Access Token)

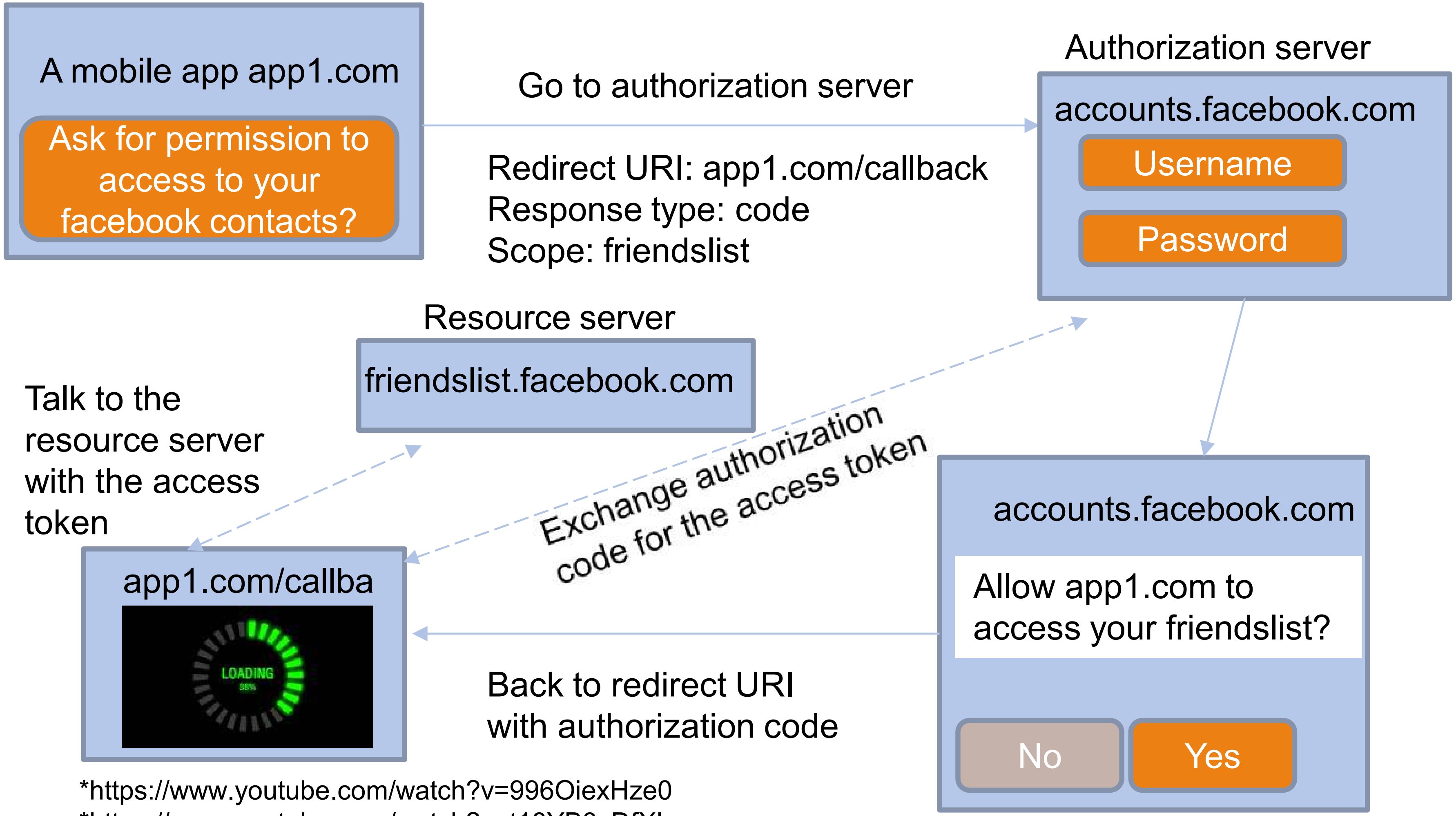
OpenID Connect (Authentication)	OAuth 2.0 (Authorization)
• Logging user in (SSO) • Making your accounts available in other systems	• Getting access to your API • Getting access to user data in other systems

An example OAuth 2.0 scenario

- You allow a mobile app to send a “Merry Christmas message” to your Facebook friends on behalf of you.
- The mobile must get access to your friends list on Facebook
- Instead of giving the mobile app your Facebook username and password, you can give the mobile app a key/access token that gives it specific permissions to get access to your Facebook friends list.

* https://developers.facebook.com/docs/facebook-login/auth-vs-data

OAuth 2.0 Code Flow*



A mobile app app1.com

Ask for permission to access to your facebook contacts?

Go to authorization server

Redirect URI: app1.com/callback
Response type: code
Scope: friendslist

Authorization server

accounts.facebook.com

Username

Password

Resource server

friendslist.facebook.com

Talk to the resource server with the access token

Exchange authorization code for the access token

accounts.facebook.com

Allow app1.com to access your friendslist?

No

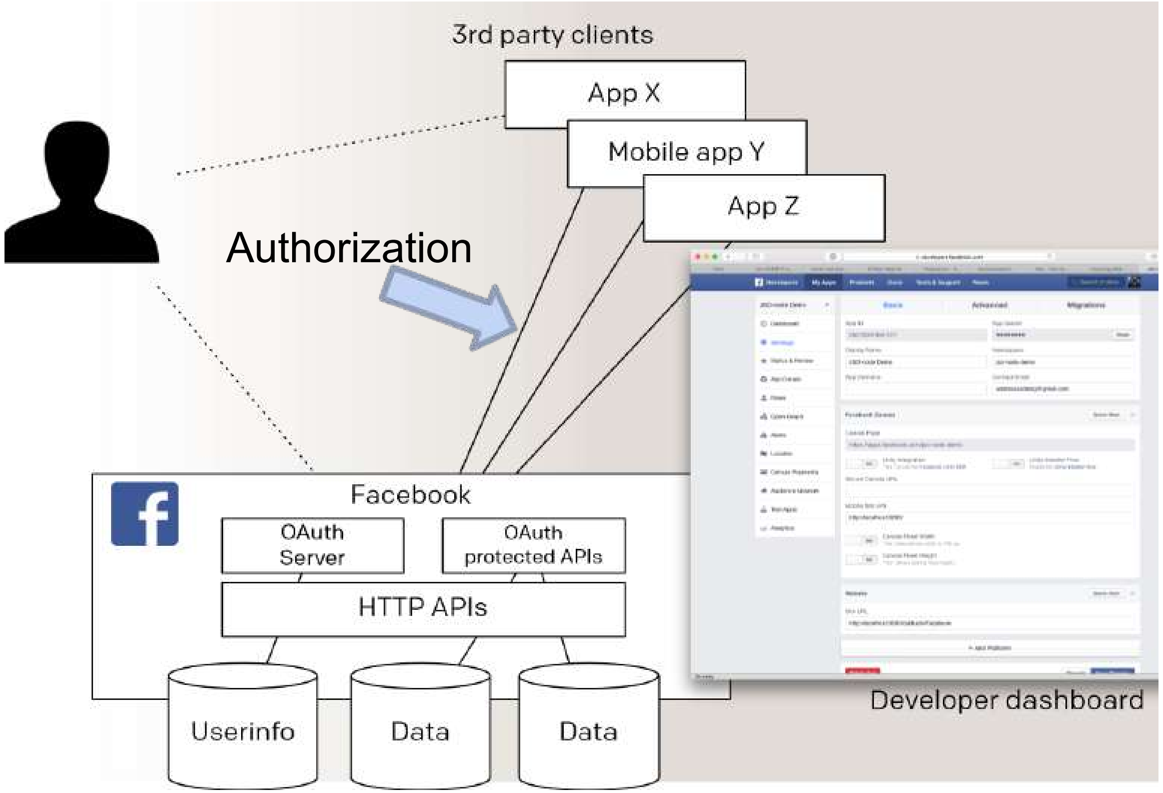
Yes

app1.com/callba

Back to redirect URI with authorization code

*https://www.youtube.com/watch?v=996OiexHze0
*https://www.youtube.com/watch?v=t18YB3xDfXI

OAuth 2.0



You give one application permission to access your data in another application.

Control hijacking

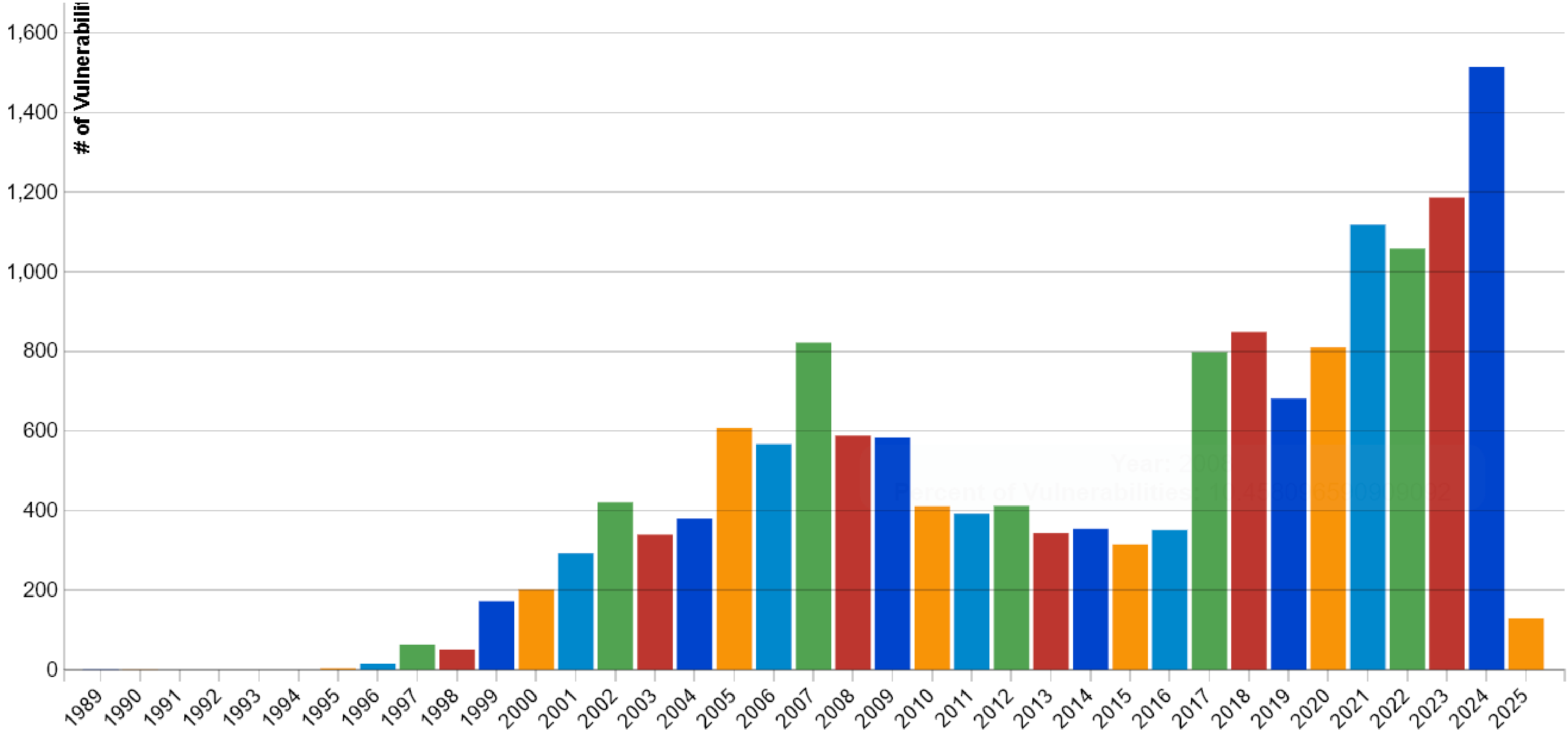
Control hijacking

- Attacker’s goal
  - Take over target machine (e.g., webserver)
  - Execute arbitrary code on target by hijacking application control flow
  - Compromise
    - Confidentiality, Integrity, Availability
- Targets mainly C/C++ code

Buffer overflow attacks

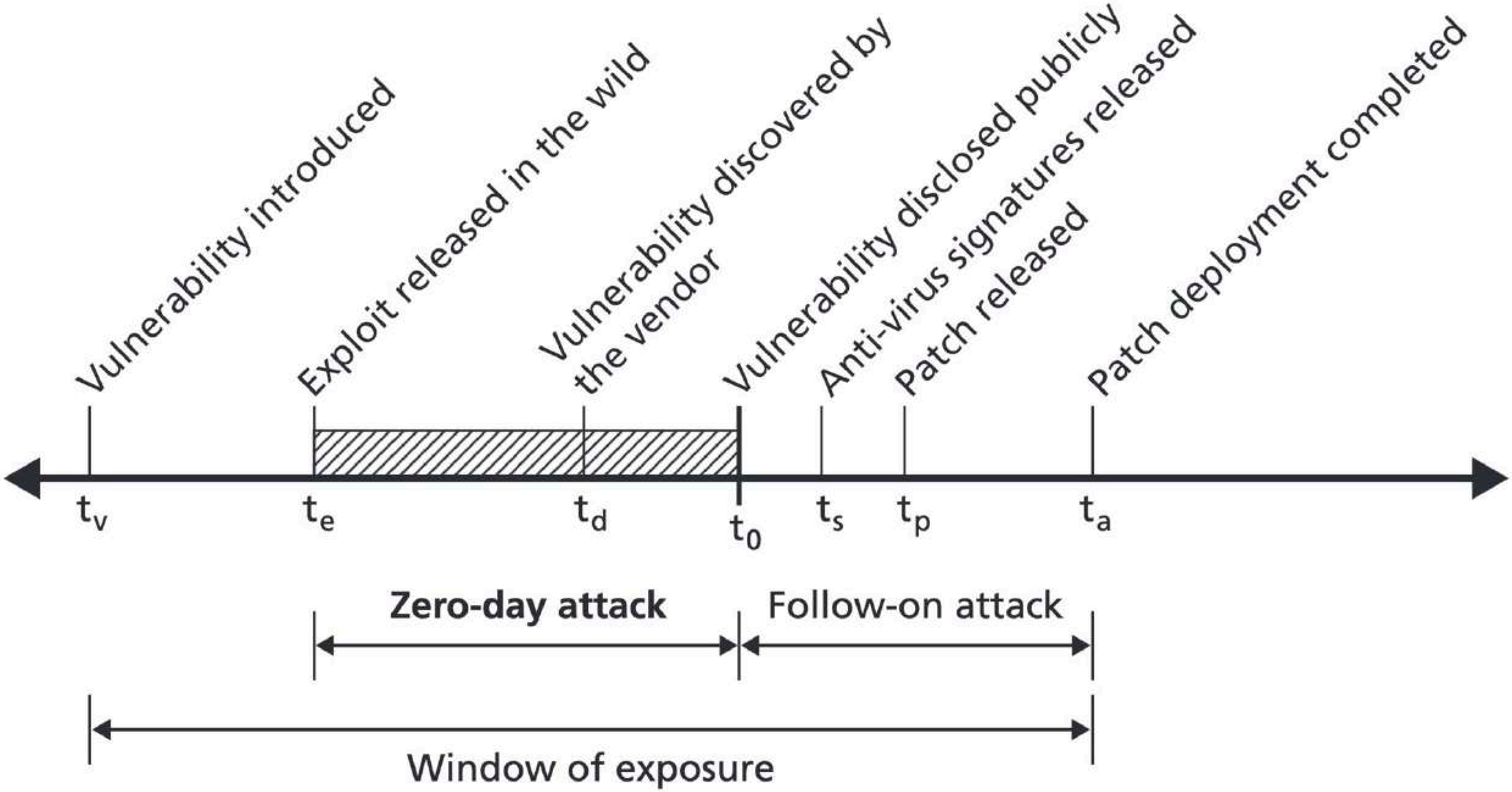
- *Morris worm* - fingerd on VAXes(1988)
- *CodeRed* - MS IIS Web Server(2001)
- *SQL Slammer* - MS SQL Server (2003)
- *Heartbleed* - OpenSSL and Secure Web Servers (2014)
- Google Chrome Heap Buffer Overflow (2023)

Buffer overflow vulnerabilities



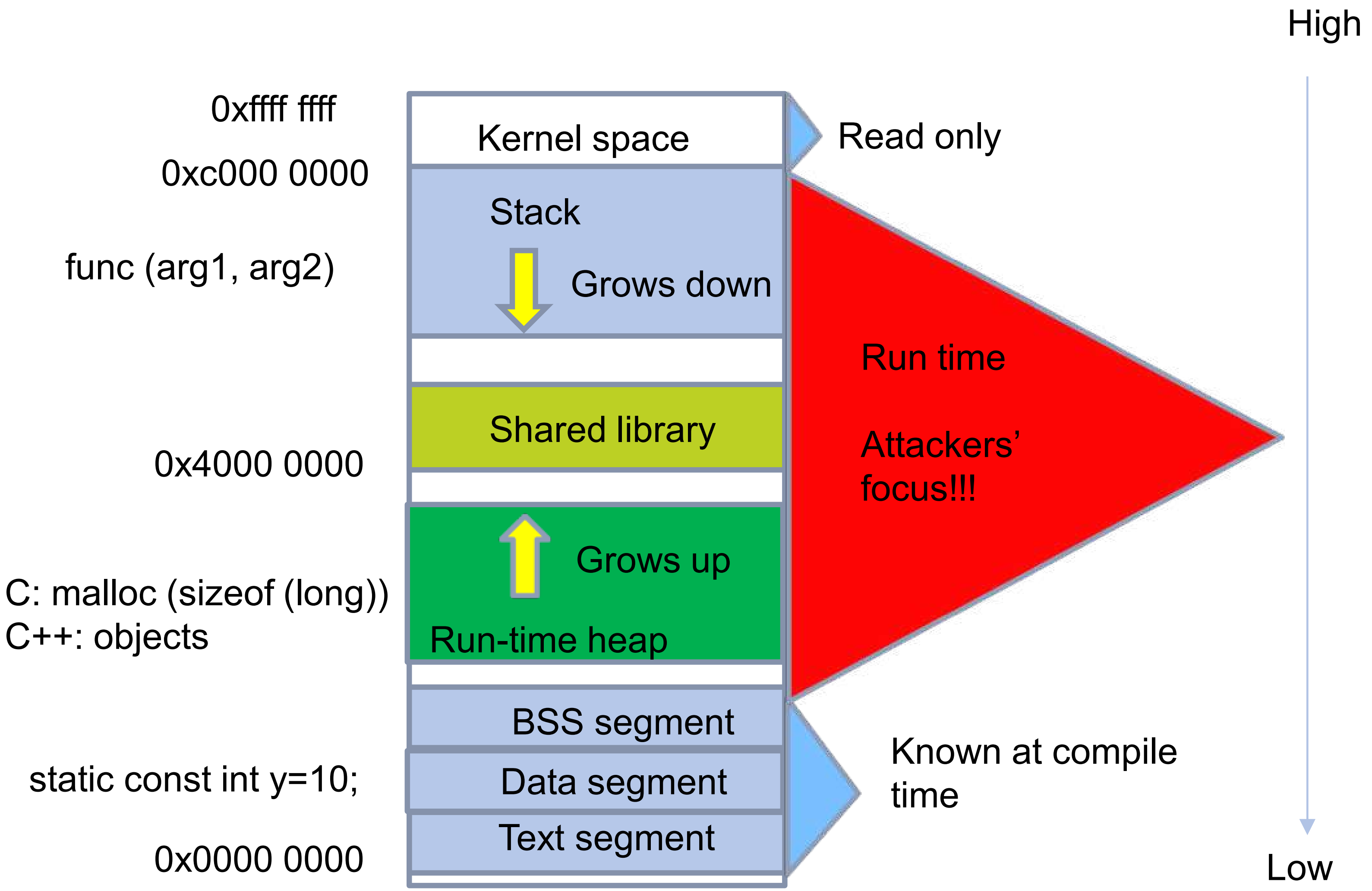
https://nvd.nist.gov/vuln/search/statistics?form_type=Basic&results_type=statistics&query=buffer+overflow&search_type=all&isCpeNameSearch=false

Zero day vulnerabilites & exploits


Vulnerability introduced
Exploit released in the wild
Vulnerability discovered by the vendor
Vulnerability disclosed publicly
Anti-virus signatures released
Patch released
Patch deployment completed
t_v
t_e
t_d
t_0
t_s
t_p
t_a
Zero-day attack
Follow-on attack
Window of exposure

https://en.wikipedia.org/wiki/Zero-day_vulnerability#/media/File:Vulnerability_timeline.png

Linux process memory layout



High

0xffff ffff

Kernel space

Read only

0xc000 0000

Stack

func (arg1, arg2)

Grows down

Run time

Attackers' focus!!!

Shared library

0x4000 0000

Grows up

C: malloc (sizeof (long))
C++: objects

Run-time heap

BSS segment

Data segment

static const int y=10;

Text segment

0x0000 0000

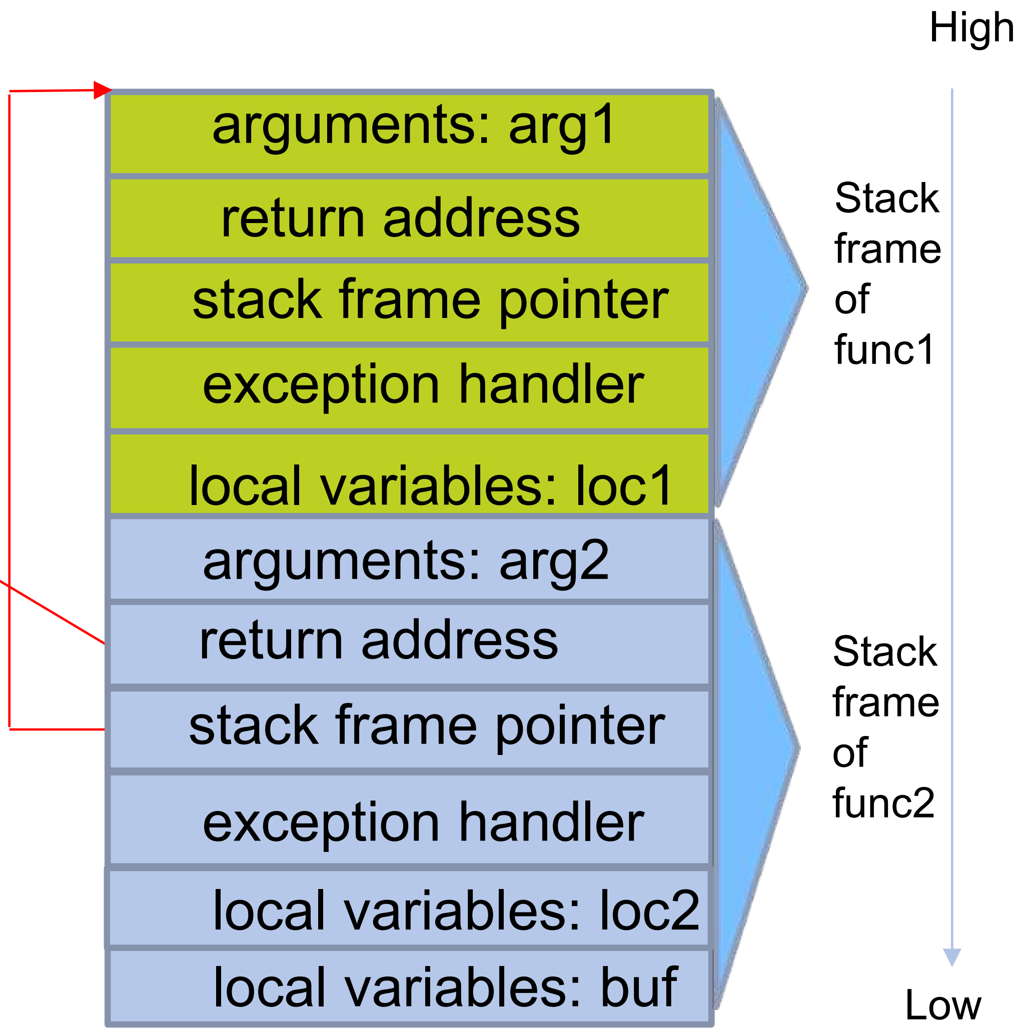
Known at compile time

Low

Stack and function calls

```
void func1(char *arg1)
{
  int loc1;
  func2(arg1);
  …
}

void func2 (char *arg2)
{  int loc2;
    char buf[128];
    strcpy(buf, arg2)
}
```



High

arguments: arg1

return address

stack frame pointer

exception handler

local variables: loc1

Stack frame of func1

arguments: arg2

return address

stack frame pointer

exception handler

local variables: loc2

local variables: buf

Stack frame of func2

Low

What are stack overflows?

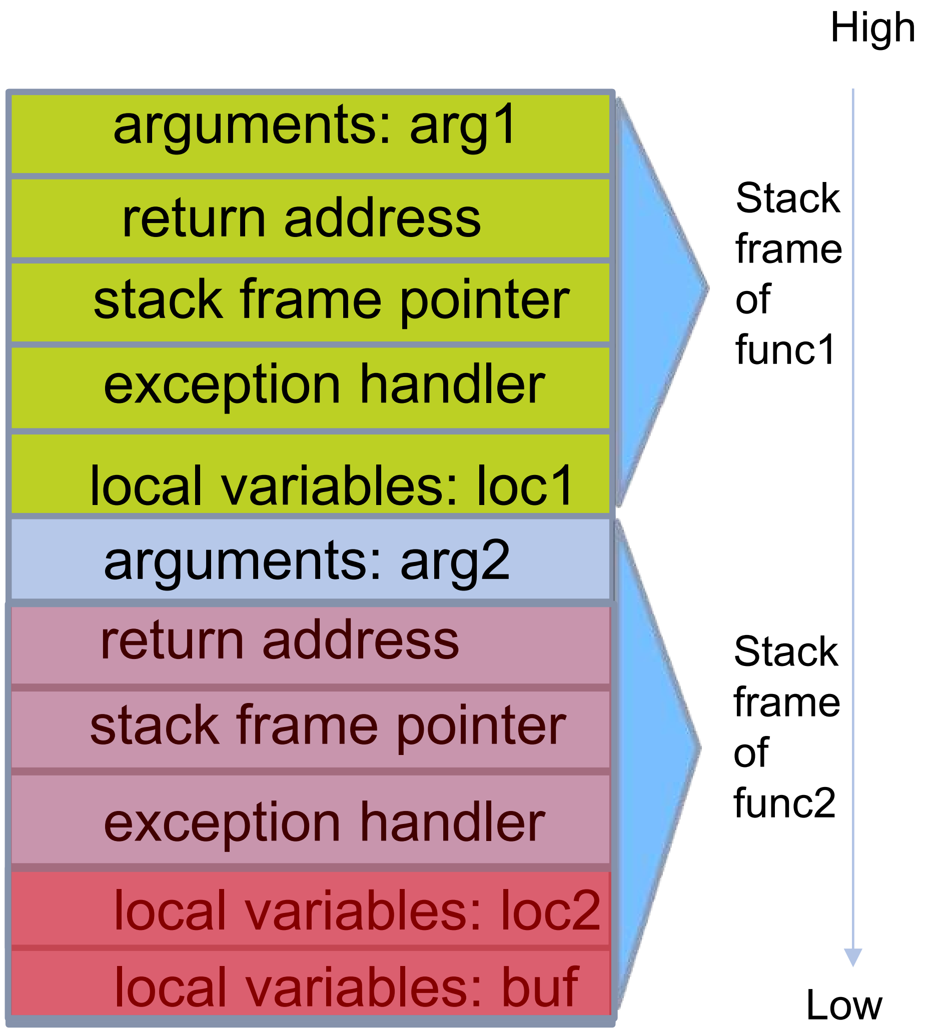
```
void func2 (char *arg2)
{  int loc2;
   char buf[128];
   strcpy(buf, arg2)
}
```

Problem: no length checking in strcpy()

What if *arg2 is > 128 bytes long?

Buffer can overflow

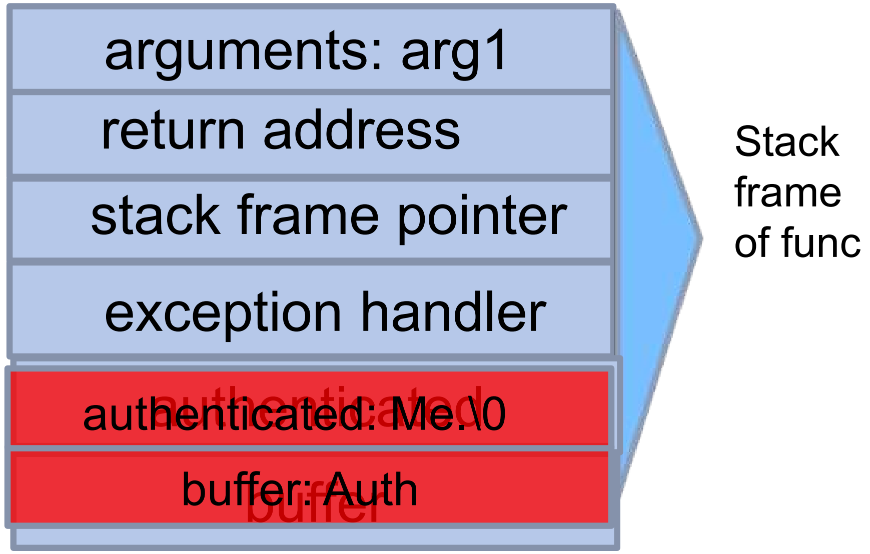
- Other local variables
- Exception handler
- Return address



Corrupt control logic using stack overflow

```
int main ()
{  char mystr[10];
   fgets(mystr, sizeof(mystr), stdin);
   func(mystr);
   …
}

void func (char *arg1)
{ int authenticated = 0;
   char buffer[4];
   …
   (some authentication check code here
    to set value 1 or 0 to variable authenticated
    Correct  Username&Passwd, assign value 1 to
                  authenticated
     Wrong Username&Passwd, assign value 0 to
                    authenticated)
   …
   strcpy(buffer, arg1);
   if(authenticated) { some critical operation…}
}
```

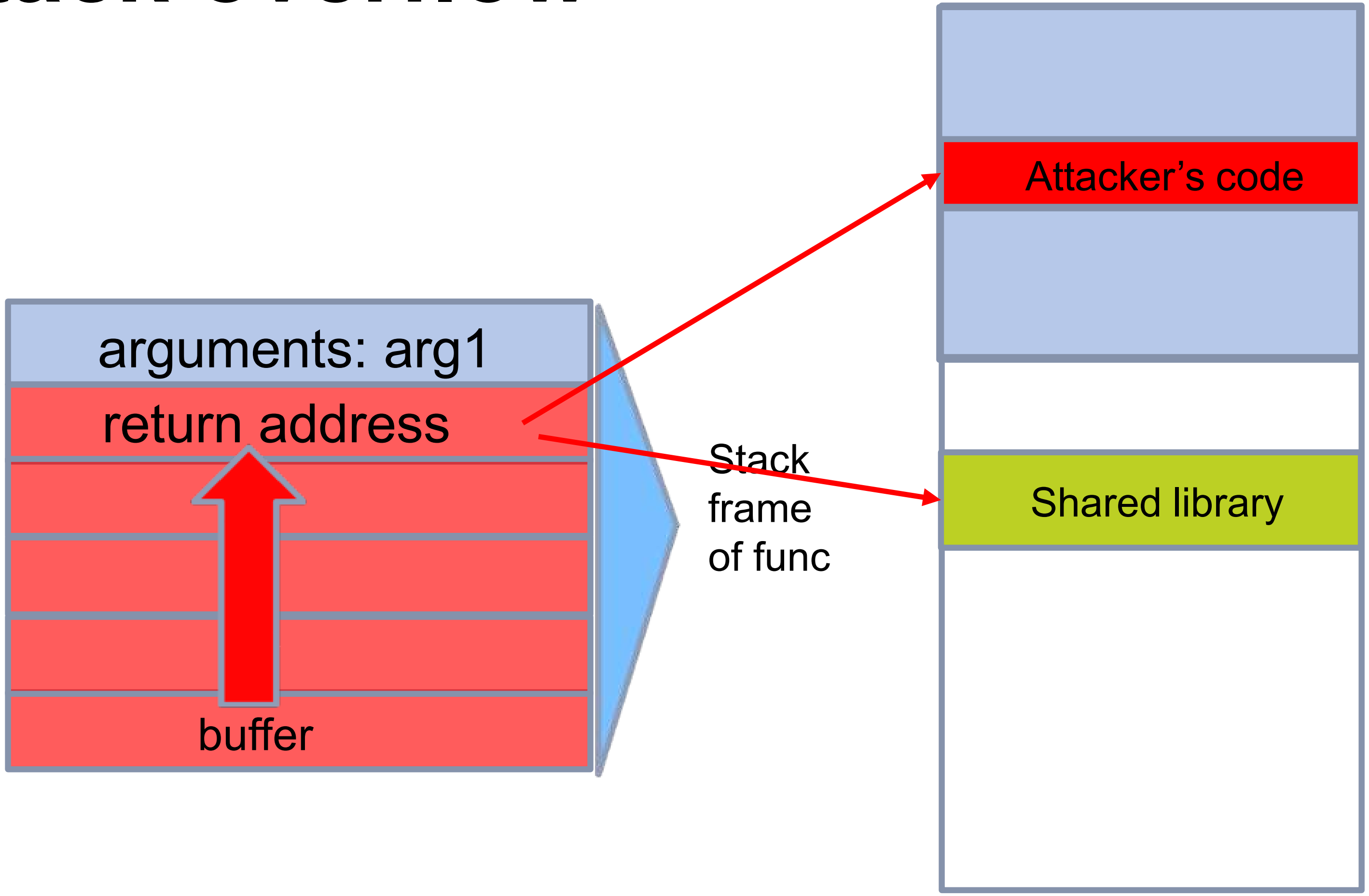


authenticated = 4d 65 2E 00 (Me.\0) != 0

Attacker types in:
“AuthMe.”;

Authentication check result bypassed

Run code of attacker’s choice using stack overflow



Steal information using stack read overflow

```
int main()
{
   char buf [128];
   …
  for(int i= 0; i<length; i++ )
 {
    putchar( buf[i]);
  }
   …
}
```

The value of "length" is not checked. The value (e.g., 138) may exceed the actual length of the buffer.

- Heartbleed was a read overflow attack
- The SSL server should accept a "heartbeat" message that it echoes back
- The heartbeat message specifies the length of the message to echo back However, SSL software did not check the length
- Attacker requests a longer length and reads past the content of the buffer.

Defend against buffer overflow

- Always use safe functions
  - Unsafe functions
    - strcpy(char * dest, const char * src)
    - strcat(char * dest, const char* src)
    - gets(char *)
    - strncpy(char * destination, const char * source, size_t num)
    - …
  - Safe functions are functions that
    - Check the length of the inputs
    - Ensure proper termination of the string
    - E.g., secure Windows c run-time libraries

```
errno_t strcpy_s (
      char *strDestination,
      size_t numberOfElements,
      const char *strSource
);
```

Strncpy does not terminate string with NULL

```
char str[3];
strncpy(str, “bye”, 3);
int x = strlen(str);
```

x can be longer than 3.
Can lead to read overflow attack, i.e., attackers can read more than str until a NULL is met

Defend against buffer overflow (cont')

- Leverage defences in compilers, e.g.,
  - GCC (*-fstack-protector*)
  - Windows Visual studio
    - E.g., /GS option, /SAFESEH option, /SEHOP option
- Check length when read/write buffer
- Use tools to audit source code
  - E.g., static code analysis (later lecture, stay tuned…)
- Rewrite software in type-safe language

Why type-safe language helps*?

- Python

```
>>> mystring="This is my string"
>>> print mystring
 This is my string
```

- C

```
char mystring[20]=“This is my string”;
printf(“%s”, mystring);
```

- Type-safe:
  - Python, Java, Ruby, Go, C#, Javascript, Smalltalk, Haskell, Scheme, Ada, …

You don’t have to specify how big your string will be.

All you do is to assign a string to your variable and the Python language takes care of the rest for you.

The programmer is responsible for defining both what the variable will store and what the size of the variable in memory will be.

If the programmer allocates 20 bytes of memory then tries to store 30 bytes, a buffer overflow happens.

*https://isc.sans.edu/forums/diary/A+buffer+overflow+in+a+Type+safe+Language/17749/

Next week:

Security engineering book (Ross):

- Chapter 2: Who is the opponent
- Chapter 27.3: Lessons from safety-critical systems

The threat modeling manifesto:
https://www.threatmodelingmanifesto.org/

OWASP TG:
2.5 Threat modeling
https://owasp.org/www-project-web-security-testing-guide/v42/2-Introduction/README#Threat-Modeling