DI-FCT-UNL
Segurança de Redes e Sistemas de Computadores

Mestrado Integrado em Engenharia Informática
2º Semestre

- *Access Control*

Outline

- **Access control topics**
  - Principles of Access Control Models: Subject, Objects and Permissions (Access-Rights or Authorizations)
  - Access Control Policy Models: MAC, DAC, RBAC, ABAC
  - Mandatory Access Control (MAC)
  - Discretionary Access Control (DAC)
    - Case: Unix File System
  - Role Based Access Model (RBAC)
    - Example: RBAC in a Banking System
  - Attribute-Based Access Control (ABAC)
  - Complementary related topics
    - Identity, Credentials and Access Management
    - Trust framework for access control enforcement and authorization management

Outline

- **Access control topics**
  - Principles of Access Control Models: Subject, Objects and Permissions (Access-Rights or Authorizations)
  - Access Control Policy Models: MAC, DAC, RBAC, ABAC
  - Mandatory Access Control (MAC)
  - Discretionary Access Control (DAC)
    - Case: Unix File System
  - Role Based Access Model (RBAC)
    - Example: RBAC in a Banking System
  - Attribute-Based Access Control (ABAC)
  - Complementary related topics
    - Identity, Credentials and Access Management
    - Trust framework for access control enforcement and authorization management

Access Control

- "The prevention of unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner"

- Or (as defined in RFC 4949): "Measures that implement and assure security services in a computer system, particularly those that assure access control service."

- A central element of computer security
  - Related to the materialization of the Access-Control Security Property
  - (Remember the OSI X.800 Framework and Security Services and Mechanisms Typology)

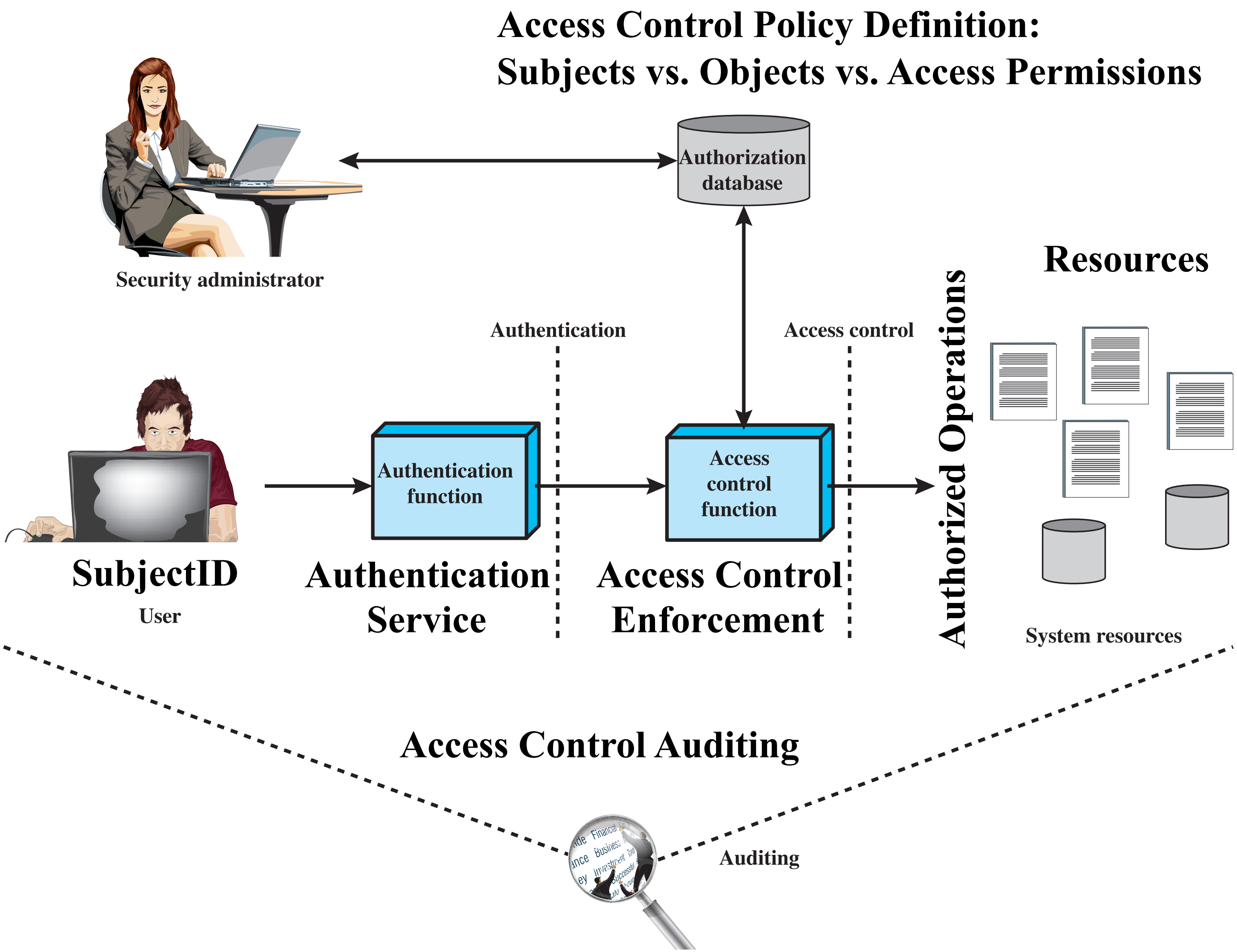
Access Control: Assumptions

- Assume principals or users (Principals – PrincipalIDs, SubjectIDs, UserIDs, …) and groups (aggregated Principals as GroupIDs)
  - Authenticate to system
    - Access control is applied over (supposed) authenticated subjects or principals
      - Relates to the need of Authentication Service
    - **But … authentication and access control are two different services (separation of concerns) using different mechanisms !!!**
- Access control services: assignment of access rights (or permissions) to access certain resources on system and their control

Access Control: Assumptions

- Access control services: assignment of **access rights (or permissions)** to access certain resources on system and their control
    - **Permission to access a resource** is also called **authorization**
    - An Access Control Service requires the definition of Access Control Policy
    - Verification and enforcement via an **Access-Control Service Reference Monitor**
        - Set and verification of access control enforcements (as access-control definitions) providing the related control guarantees
        - **Access-Control Reference Monitor: a trusted process that verify/ monitors/apply access control enforcements**
            - » **allowing or denying the access**
            - » **for the execution of specific operations (OPi) on resources (Rj) intended by well-defined (and previously authenticated) principals (SujectIDk)**

Access Control Principles



Access Control Elements

- Subjects (or Principals), Objects and Access Rights (or Permissions)

Subject

An entity capable of accessing objects

Ex. of classes:
- Object Owner
- Group
- World (All)

Object

A resource to which access is controlled

Entity used to contain and/or receive information

Files – Data or Binary Files, DIRs,
Data Records,
KVS entries,
DB Tables, Columns, Lines, …
Devices ,…

Access right (Permission)

Describes the way in which a subject may access an object

Operations, could include:
- Read
- Write
- Execute
- Delete
- Create
- Search

Ex: Concretization

Permission Grain Specification on Access Control

- Important issue: limitations of coarse-grain access control enforcements
  - Devices / Sensors / Data in smartphones, tablets (ex., Android Access Control Ecosystem)
    - Two only permissions: ALL or NOTHING
    - What about the SubjectID / eUID
      - Only one user: USER is also the SYS ADMIN
      - Can do everything ! She/he installs and Executes everything
      - What about user authentication ?
    - Access Control Monitoring at Middleware Level (Out of the Base OS Foundations)
    - What about App Sandboxing Protection ?
    - What about Access Control Auditing and Awareness ?

Problems in current smartphones, tablets …

- Problem: Are current smartphones/tablets ready to be used in BYOD paradigms, running sensitive and no-sensitive apps in the same execution eco-system ?
- No ! Different issues involved, but access control is one of the most prevalent problems
    - Lack of appropriate TRUSTED EXECUTION ENVIRONMENT and complete approach of Access Control System Design Principles
    - Lack of Fine-Grain Access Control
    - No separation of roles: SYSADMIN and USER
    - Too High Level Trust Computing Base Assumptions:
        - Ex. in ANDROID Devices: OS, Device Drivers, Delvik VM, Application Level Support Libraries
    - A situation where "the user … can be easily "the adversary" !

Violation of Least-Privilege Assumptions

- Important issue: limitations of coarse-grain access control enforcements and the privilege escalading problem
- Consequences:
    - Lots of access control problems … (more on this later)
    - Confused Deputy Problem: a computer program that is innocently fooled by some other party into misusing its authority.
        - Ex., Use of the Video Camera. Microphone, GPS location, SD card, etc. … by a App with given authorization as resources that will be used illicitly by another installed App without authorization for that
            - » One of the more prevalent attacks on current ANDROID OS devices
        - Web Security Violation with CSRF (Cross-Site Reqiuest Forgery) and XSS (Cross-Site Scripting) Attacks
            - » One of the more prevalent attacks on Web Applications and Services
        - All are examples of the violation/limitation of the PRINCIPLE OF THE LEAST PRIVILEGE in Access Control System Design Assumptions !

Design criteria in Access Control Requirements (1)

> *See also bibliography for more details*

- Fine and coarse specifications
  - Grain of Access Control Enforcements
  - fine-grained specifications allow access regulated at the level of individual fields / records in files, etc;
  - and each individual access by a user rather than a sequence of accesses.
  - System administrators should also be able to choose coarse-grain specification for some classes of resource access.
- Principle of Least Privilege
  - it should be implemented so that each system entity is granted the minimum system resources and authorizations needed to do its work.
  - This principle tends to limit damage that can be caused by an accident, error, or unauthorized act, as a default-behavior

Design criteria in Access Control Requirements (2)

- Reliable input
  - it assumes that a user is authentic (previously authenticated); thus, an authentication mechanism is needed as a front end to an access control system.
  - Any user inputs to the access control system must also be reliable (and supposed that are inputs originated by authenticated correct users)

- Separation of duty
  - should divide steps in a system function among different individuals, so as to keep a single individual from subverting the process.

- Open vs. closed policies
  - a closed policy only allows accesses that are specifically authorized; an open policy allows all accesses except those expressly prohibited.

Design criteria in Access Control Requirements (3)

- Policy combinations, consistency and conflict resolution
  - may apply multiple policies to a *given* class of resources
  - need a procedure to resolves conflicts between policies.

- Administrative policies
  - to specify who can add, delete, or modify authorization rules, and also need access control and other control mechanisms to enforce these administrative policies.
  - A complex system or application can involve different levels of access-control policies:
    - Separation between MAC administrative policies from DAC, RBAC or ABAC policies

Outline

- **Access control topics**
  - Principles of Access Control Models: Subject, Objects and Permissions (Access-Rights or Authorizations)
  - Access Control Policy Models: MAC, DAC, RBAC, ABAC
  - Mandatory Access Control (MAC)
  - Discretionary Access Control (DAC)
    - Case: Unix File System
  - Role Based Access Model (RBAC)
    - Example: RBAC in a Banking System
  - Attribute-Based Access Control (ABAC)
  - Complementary related topics
    - Identity, Credentials and Access Management
    - Trust framework for access control enforcement and authorization management

Base Access Control Policies

- Discretionary access control **(DAC)**
  - Controls access based on the identity of the requestor and on access rules (authorizations) stating what requestors are (or are not) allowed to do
  - the data owner determines who can access specific resources.

- Mandatory access control **(MAC)**
  - Controls access based on comparing security labels with security clearances

- Role-based access control **(RBAC)**
  - Controls access based on the roles that users have within the system and on rules stating what accesses are allowed to users in given roles

- Attribute-based access control **(ABAC)**
  - Controls access based on attributes of the user, the resource to be accessed, and current environmental conditions
  - Access rights are granted to users through the use of policies which evaluate possible combined attributes (user attributes, resource attributes and environment conditions)

RBAC policy

- RBAC allows access based on the job title.
- RBAC largely eliminates discretion when providing access to objects. For example, a human resources specialist should not have permissions to create network accounts; this should be a role reserved for network administrators.

- Possible variants are sometimes defined with other designations, ex:
- RAC – Rule-Based Access Control
  - RAC method is largely context based. Example of this would be only allowing students to use the labs during a certain time of day.
- Responsibility Based Access control

ABAC policy

- An access control paradigm whereby access rights are granted to users through the use of policies which evaluate attributes (user attributes, resource attributes and environment conditions)
  – We can imagine context-aware attributed for specific ABAC models: Time-leasing conditions, Location Validity, Operation-Flow Controls, Behavioral Biometric Usage Conditions, …

Other AC policies (1)

- Possible variants are sometimes defined with other designations (classified by different authors as access control policy models). Examples include:
  - **HBAC – History Based Access Control**
    - Access is granted or declined based on the real-time evaluation of a history of activities of the inquiring party, e.g. behavior, time between requests, content of requests or state-machine of operation-flows.
  - **IBAC – Identity Based Access Control**
    - In such policies network administrators can more effectively manage activity and access based on specific individual needs.
  - **OrBAC – Organization-Based Access Control**
    - OrBAC model allows the policy designer to define a security policyfor organizational or business functions independently of the implementation. Usually, we can map on designed RBAC and ABAC restrictions

Other AC policies (2)

- Possible variants are sometimes defined with other designations (classified by different authors as access control policy models). Examples include:
  - **RAC - Rule Based Access Control**
    - RAC methods are defined largely as context based access control. Example of this would be only allowing students to use the labs during a certain time of day.
    - Some overlaps with ABAC and/or RBAC
  - **ResBAC – Responsibility Based Access Control**
    - Information is accessed based on the responsibilities assigned to an actor or a business role
    - Some overlaps ABAC and/or RBAC and/or OrBAC

Outline

- **Access control topics**
  - Principles of Access Control Models: Subject, Objects and Permissions (Access-Rights or Authorizations)
  - Access Control Policy Models: MAC, DAC, RBAC, ABAC
  - Mandatory Access Control (MAC)
  - Discretionary Access Control (DAC)
    - Case: Unix File System
  - Role Based Access Model (RBAC)
    - Example: RBAC in a Banking System
  - Attribute-Based Access Control (ABAC)
  - Complementary related topics
    - Identity, Credentials and Access Management
    - Trust framework for access control enforcement and authorization management

MAC Policy

- MAC Level Enforcement:
  - Examples:
    - Kernel-Based Mandatory Access Control
    - Only code running in supervised mode can access/manage OS system Resources
    - Code executed beyond the System Calls (Calls from running Processes)
  - In MAC, users couldn’t have much freedom to determine who has access to their own files.
  - For example, security clearance of users and classification of data (as confidential, secret or top secret) are used as security labels to define the level of trust.

MAC policy concretizations: OS (1)

- Refers to a type of access control by which an OS constrains the ability of a *subject* or *initiator* to access or generally perform some sort of operation on an *object* or *target*, directly controlled by the OS kernel in supervised running model
- In practice, a subject is usually a process or thread; objects are constructs such as files, directories, TCP/UDP ports, shared memory segments, IO devices, etc.
- Subjects and objects each have a set of security attributes.

- Operation
  - Whenever a subject attempts to access an object, an authorization rule directly defined and enforced by the operating system kernel examines these security attributes and decides whether the access can take place.
  - Any operation by any subject on any object is tested against the set of authorization rules (aka OS *policy*) to determine if the operation is allowed

MAC policy concretizations: DBMS (2)

- A DBMS (Data Base Management System) in its access control service, can also apply mandatory access control;
- Implemented by the DBMS runtime support environment
- in this case, the objects are tables, views, procedures, etc.

DAC policy

- DAC level Enforcement:
  - In DAC, the data owner determines who can access specific resources. For example, a system administrator may create a hierarchy of files to be accessed based on certain default permissions for certain users, groups of users and allowed operations. Owners can rewrite these permissions
  - Example of UNIX File system permissions:
    - Read, Write, Execute Permissions
    - Principals: Owner Principals (UserIDs), GroupIDs and All (Others)
    - DAC definitions: defined and managed by the resource owner
    - Owners can pass the owning to other principals
    - Permissions scrutiny by a Kernel-Based Access Control Monitor (running as Module in supervised mode), deciding on each operation that a process (running with a correspondent efective UID - eUID) intends to apply on a resource (files, directories, device-drivers, sockets, message-queues, etc)
      - Remember: in UNIX everything (all the resources) are accessed as "file-system descriptors)

DAC concretizations: UNIX FS

- Scheme in which an owner entity may enable another entity to access some resource to perform some operation
- Provided using an access control matrix
  - One dimension consists of identified subjects that may attempt data access to the resources
  - The other dimension lists the objects that may be accessed
- Each entry in the matrix indicates the access rights of a particular subject for a particular object

DAC and Access Control Matrix

OBJECTS

SUBJECTS		File 1	File 2	File 3	File 4
	User A	Own Read Write		Own Read Write	
	User B	Read	Own Read Write	Write	Read
	User C	Read Write	Read		Own Read Write

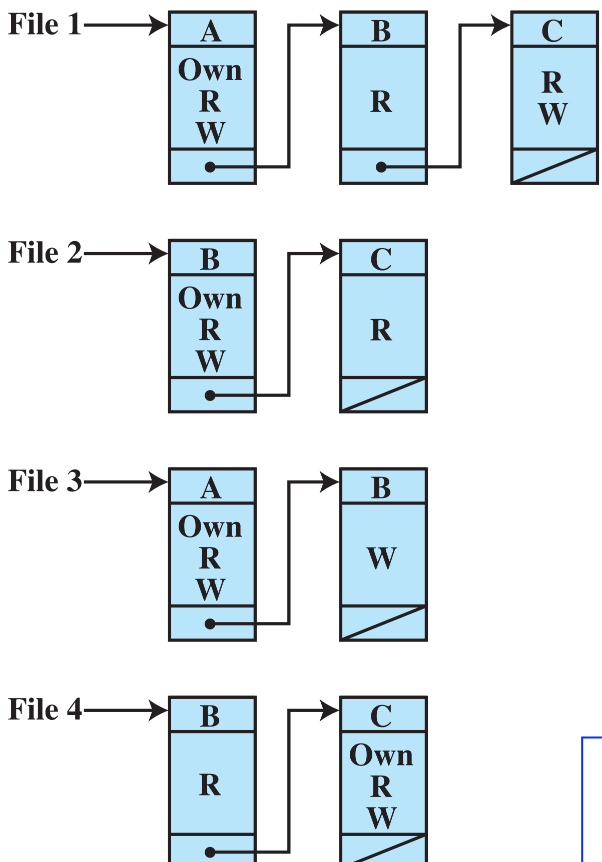
Protection Domains

- Set of objects with associated access rights
- In access matrix view, each row defines a protection domain
  - Not necessarily just a user
  - May be a limited subset of user's rights
  - Applied to a more restricted process

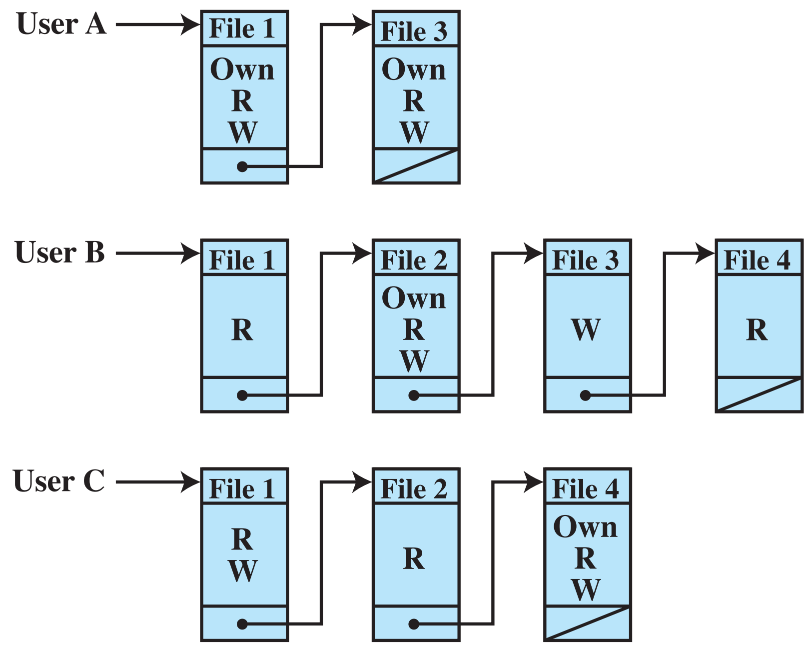
- The association between a process and a domain may be static or dynamic
  - Ex., during a process execution it may require different access rights for each procedure
  - In general: minimization of access rights overtime (controlled by protection domain)

Other Access Control Structures

Access Control Lists



Capability Lists



Example:
Object: Files
Subjects: Users

Typical Authorization Table

Subject	Access Mode	Object
A	Own	File 1
A	Read	File 1
A	Write	File 1
A	Own	File 3
A	Read	File 3
A	Write	File 3
B	Read	File 1
B	Own	File 2
B	Read	File 2
B	Write	File 2
B	Write	File 3
B	Read	File 4
C	Read	File 1
C	Write	File 1
C	Read	File 2
C	Own	File 4
C	Read	File 4
C	Write	File 4

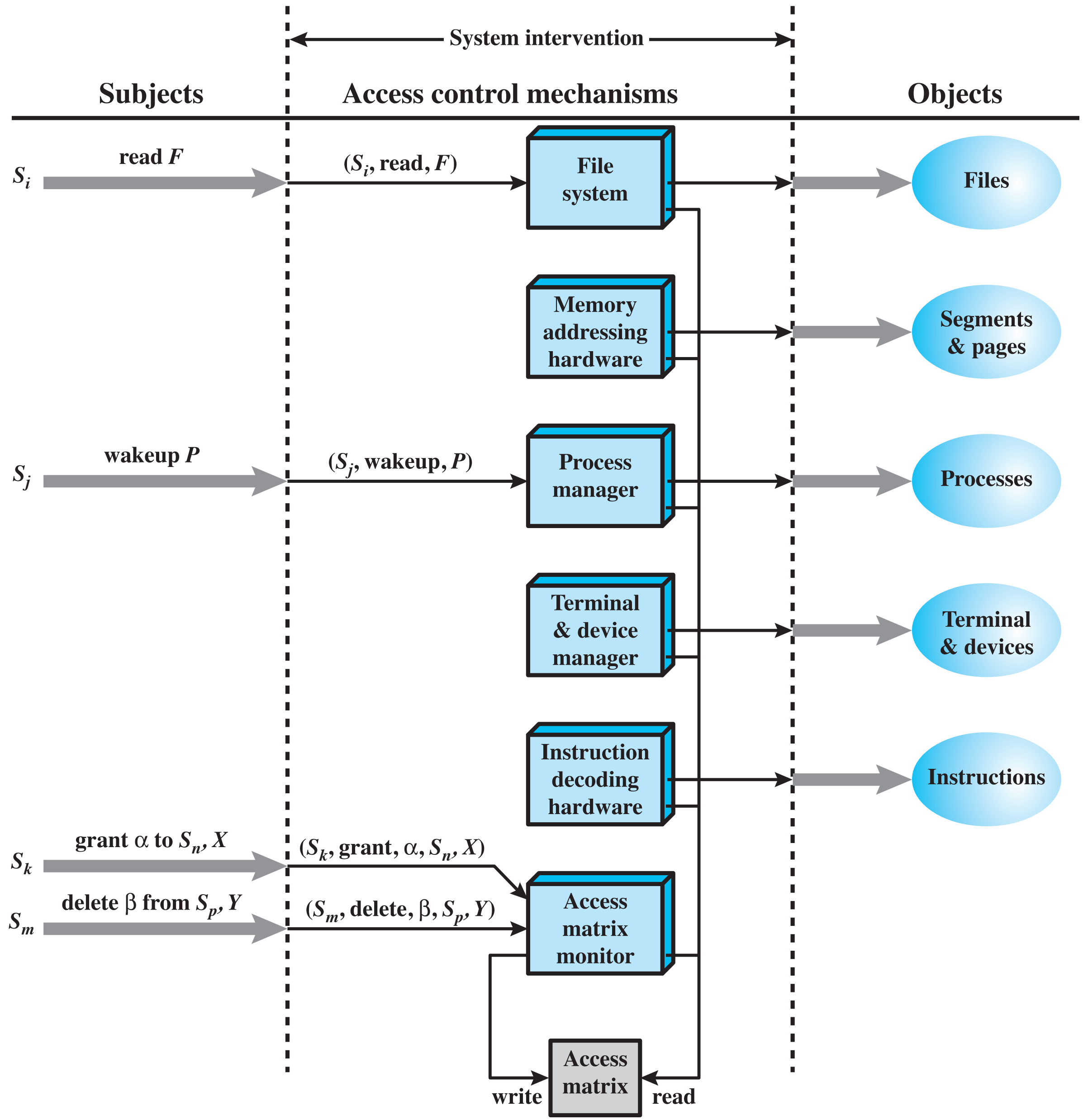
Extended Access Control Matrix

OBJECTS

		subjects			files		processes		disk drives	
		S_1	S_2	S_3	F_1	F_2	P_1	P_2	D_1	D_2
SUBJECTS	S_1	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
	S_2		control		write *	execute			owner	seek *
	S_3			control		write	stop			

* - copy flag set

Access Control Function



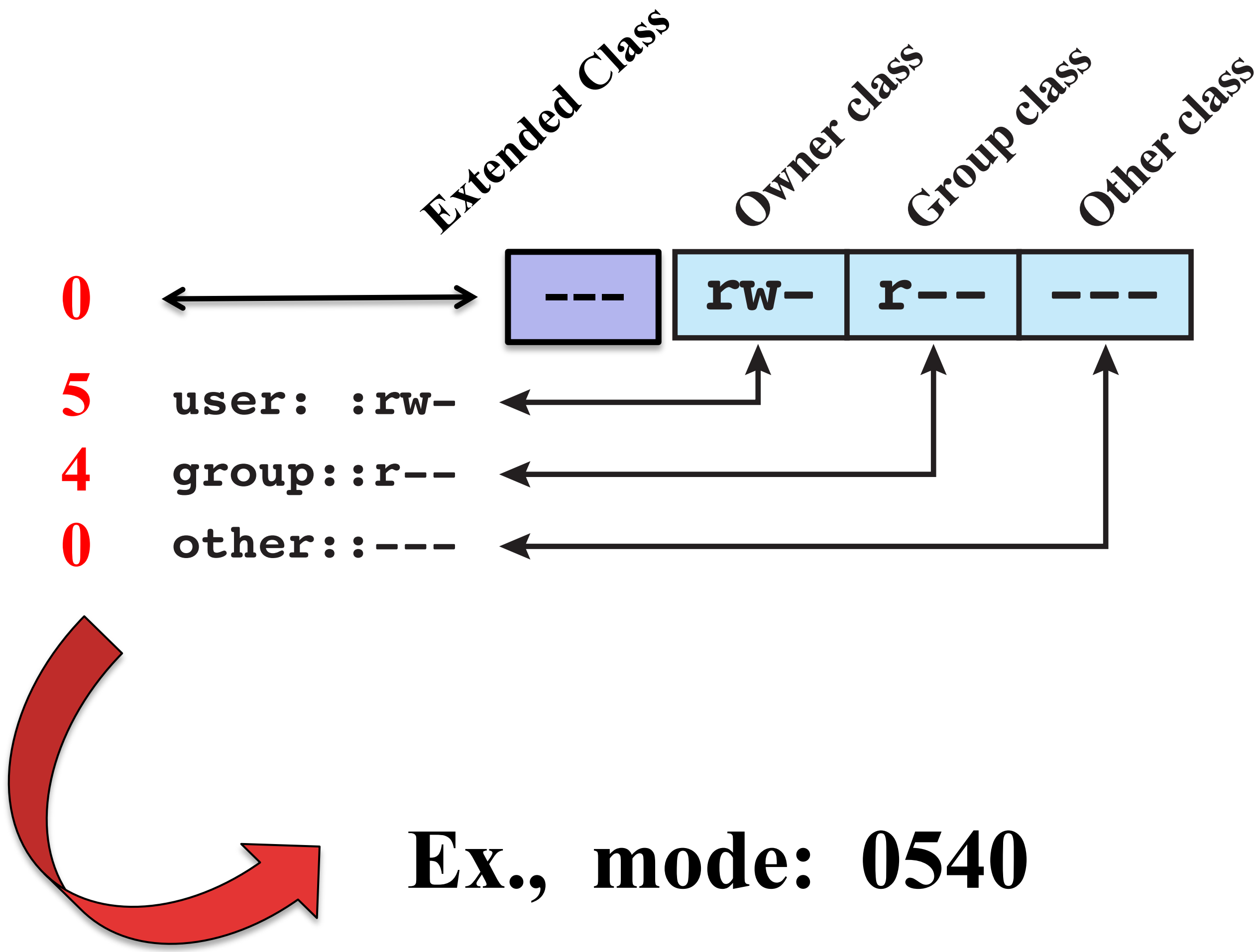
DAC and UNIX File System Concepts

- UNIX files administered using inodes
  - Control structure with key info on file
    - Attributes, permissions of a single file
  - May have several names for same inode
  - Have inode table / list for all files on a disk
    - Copied to memory when disk mounted

- Directories form a hierarchical tree
  - May contain files or other directories
  - Are a file of names and inode numbers

UNIX File Access Control

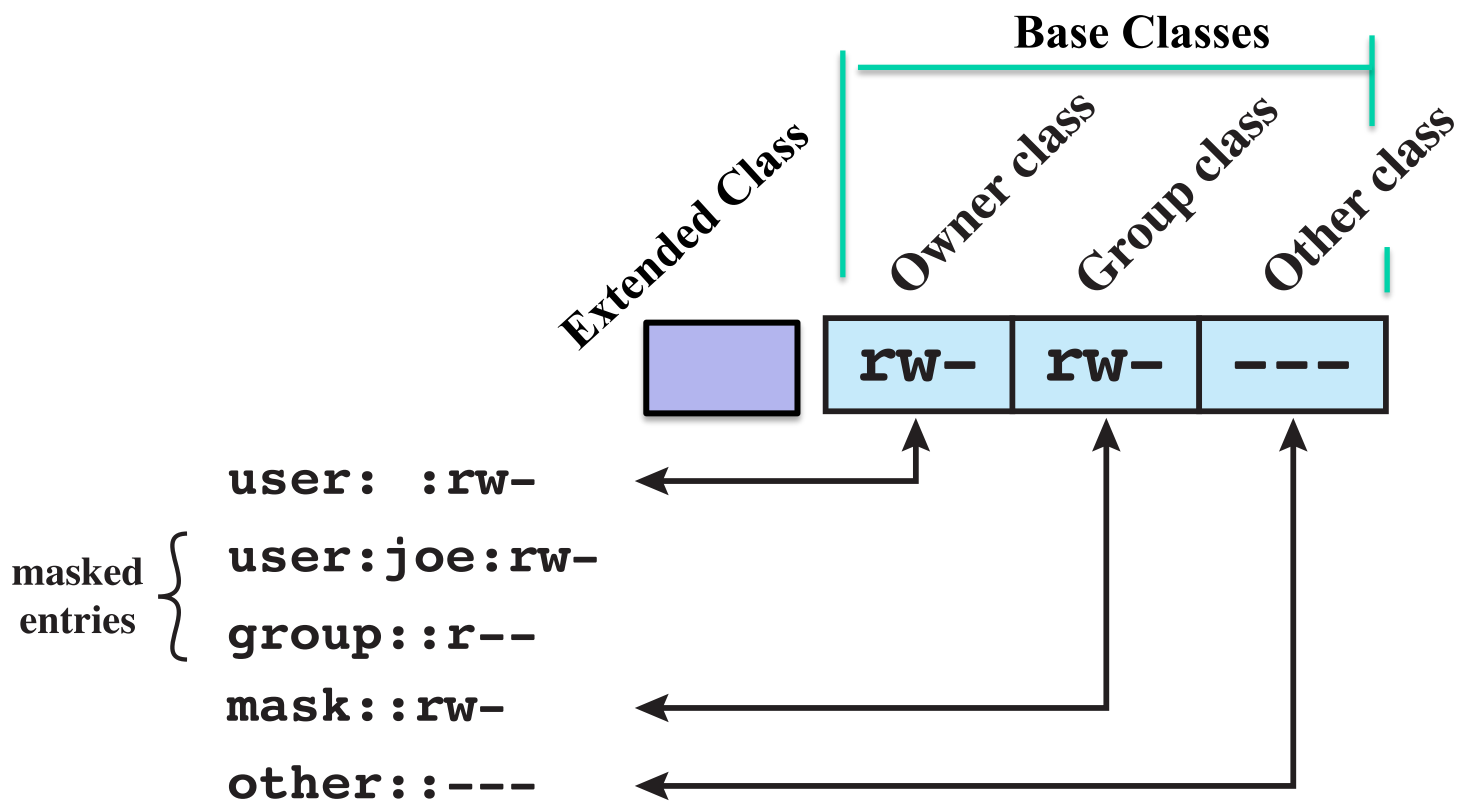
- Expression of DAC in the UNIX File System



Ex., mode: 0540

Extended File Access Control

- Expression of DAC in the UNIX File System



UNIX File Access Control

- "set user ID"(SetUID) or "set group ID"(SetGID)
  - The system temporarily uses rights of the file owner / group in addition to the real user's rights when making access control decisions
  - Enables privileged programs to access files / resources not generally accessible
- Sticky bit
  - on directory limits rename/move/delete to owner
- Superuser
  - is exempt from usual DAC restrictions

Examples

- See *chown* and *chgrp* in UNIX file system
- **chown** -- change file owner and group
  – **chown** [**-fhv**] [**-R** [**-H** | **-L** | **-P**]] owner[:group] file ...
  – **chown** [**-fhv**] [**-R** [**-H** | **-L** | **-P**]] :group file …
- **chgrp** -- change group
  – **chgrp** [**-fhv**] [**-R** [**-H** | **-L** | **-P**]] group file …

Examples

- See *chmod* in UNIX file system
  - **chmod** [**-fv**] [**-R** [**-H** | **-L** | **-P**]] mode file ...
  - **chmod** [**-fv**] [**-R** [**-H** | **-L** | **-P**]] [-a | +a | =a] ACE file ...
  - **chmod** [**-fhv**] [**-R** [**-H** | **-L** | **-P**]] [**-E**] file ...
  - **chmod** [**-fhv**] [**-R** [**-H** | **-L** | **-P**]] [**-C**] file ...
  - **chmod** [**-fhv**] [**-R** [**-H** | **-L** | **-P**]] [**-N**] file ...
- Access control modes (can combine them):
  - Modes: 4000, 2000, 1000
    - for setting eUID on owner, group and sticky-bit respectively
  - Modes: 0400, 0200, 0100 for w r x to the owner
  - Modes: 0040, 0020, 0010 for w r x to the group
  - Modes: 0004, 0002, 0001 for w r x for others

Extensions: UNIX Access Control Lists

- Many UNIX-based distributions support ACLs as extended mechanism
  - Can specify any number of additional users / groups and associated rwx permissions
  - ACLs are optional extensions to the standard permissions
  - Group permissions also set max ACL permissions
- When access is required
  - Select most appropriate ACL
    - owner, named users, owning / named groups, others
  - Check if have sufficient permissions for access

MAC and DAC Enhanced Linux Distributions

- See more on different MAC evolved mechanisms for security enhanced implementations on UNIX/LINUX distributions, SUSE Linux-App Armor, Tomoyo Linux, Trusted Solaris, Windows (since 2008), Mac OS-X and others
- Ex., summary on:
- https://en.wikipedia.org/wiki/Mandatory_access_control

Effectiveness of Access Control Policies

- Dependence from the Authentication Procedure (Authentication Service)
- Proper access control enforcements must be applied to "Authenticated" entities
  - In the context of "authenticated principals in sessions, where operations and access to objects/resources will be done"
  - Need to control such sessions (established on authentication proofs of principals involved)
- Two concerns must be carefully addressed:
  - Prevention/avoidance of "Broken Authentication and Session Management" vulnerabilities
  - Broken Access Control vulnerabilities

Unfortunately … two major vulneabilities found in practice (see, for example, Web Authentication (Un)Security, ex., https://www.owasp.org/index.php/Top_10_2017-Top_10 (2nd and 4th more vulnerable issues in today's web app@services programmiing)

Broken Authentication and Session Management

- Application functions related to authentication and session management are often implemented incorrectly
  - Allowing attackers to compromise passwords, keys, or session tokens
  - Allowing to exploit other implementation flaws to assume other users' identities (temporarily or permanently).

- What are the main causes ?

Broken Authentication and Session Management

- Causes in Web App./services, or WS Environments:
  - User authentication credentials aren't properly protected when stored using secure hashed and/or encrypted transformations. See also sensitive data exposures
    - https://www.owasp.org/index.php/Top_10_2017-A6-Sensitive_Data_Exposure
  - Credentials easily guessed or overwritten through "weak account management functions" (e.g., account creation/attributes registration change/recover passwords, weak (not authenticated) session IDs, cookies, tokens, …).
  - Session IDs exposed in the URL (e.g., allowing "over-the-shoulder" attacks and/or easy URL rewriting attacks).
  - Session IDs are vulnerable to session-fixation attacks
  - Session IDs without timeouts, or user sessions or authentication tokens (particularly single sign-on (SSO) tokens) not properly invalidated during logout procedures
  - Session IDs aren't rotated after successful login.
  - Passwords, session IDs, and other credentials are sent over unencrypted connections / unsecure channels or "apparently secure channels with many security mismatches". See "Sensitive data exposures" and also
    - https://www.owasp.org/index.php/Top_10_2017-A5-Security_Misconfiguration

Broken Authentication and Session Management

- Causes in Web App./services, or WS Environments:
  - See important practical guidelines in the OWASP ASVSP: Application Security Verification Standard Project

https://www.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project
https://www.owasp.org/images/3/33/OWASP_Application_Security_Verification_Standard_3.0.1.pdf

See also: Session Fixation Attacks

- Attacks allowing an attacker **to hijack a valid user session**, exploring a limitation in the way the web app. manages the session ID
- When authenticating a user, it doesn't assign a new session ID, making it possible to use an existent session ID.
  - The attack consists of obtaining a valid session ID (e.g. by connecting to the application), inducing a user to authenticate himself with that session ID, and then hijacking the user-validated **session by the knowledge of the used session ID**. The attacker has to provide a legitimate Web application session ID and try to make the victim's browser use it.
  - Attack Techniques (can be combined with XSS Attacks):

    see https://www.owasp.org/index.php/Session_fixation

Broken Access Control vulnerabilities

- Consider the types of authorized users of your system. Are users restricted to certain functions and data? Are unauthenticated users allowed access to any functionality or data?
- Exploits: Attackers, who are authorized users, simply change a parameter value to another resource they aren't authorized for. Is access to this functionality or data granted?

See:

- https://www.owasp.org/index.php/Top_10_2017-A4-Broken_Access_Control

Correct Approach:

- **Check access + Use per user or session indirect object references + Automated verification**
- Use of REFERENCE MONITORS for AUDITABLE and CONTROLLED ACCESS-CONTROL POLICY ENFORCEMNTS

  Nothing can be done, without the scrutiny of this reference monitor that must "attest" validations for the correct access.

  => Used as a central /auditable management of authorization policy enforcement

Example: HTTP Base Authentication

- The HTTP Base Authentication Protocol is a typical example extending the permissions of file-access for "remote" HTTP use
  - Notice: the Web (http) Server runs locally with certain DAC access control modes
  - You must avoid to put such servers running with eUID root or root owner …. Why ?
  - You must avoid also unnecessary "highest" access control privileges … Why ?

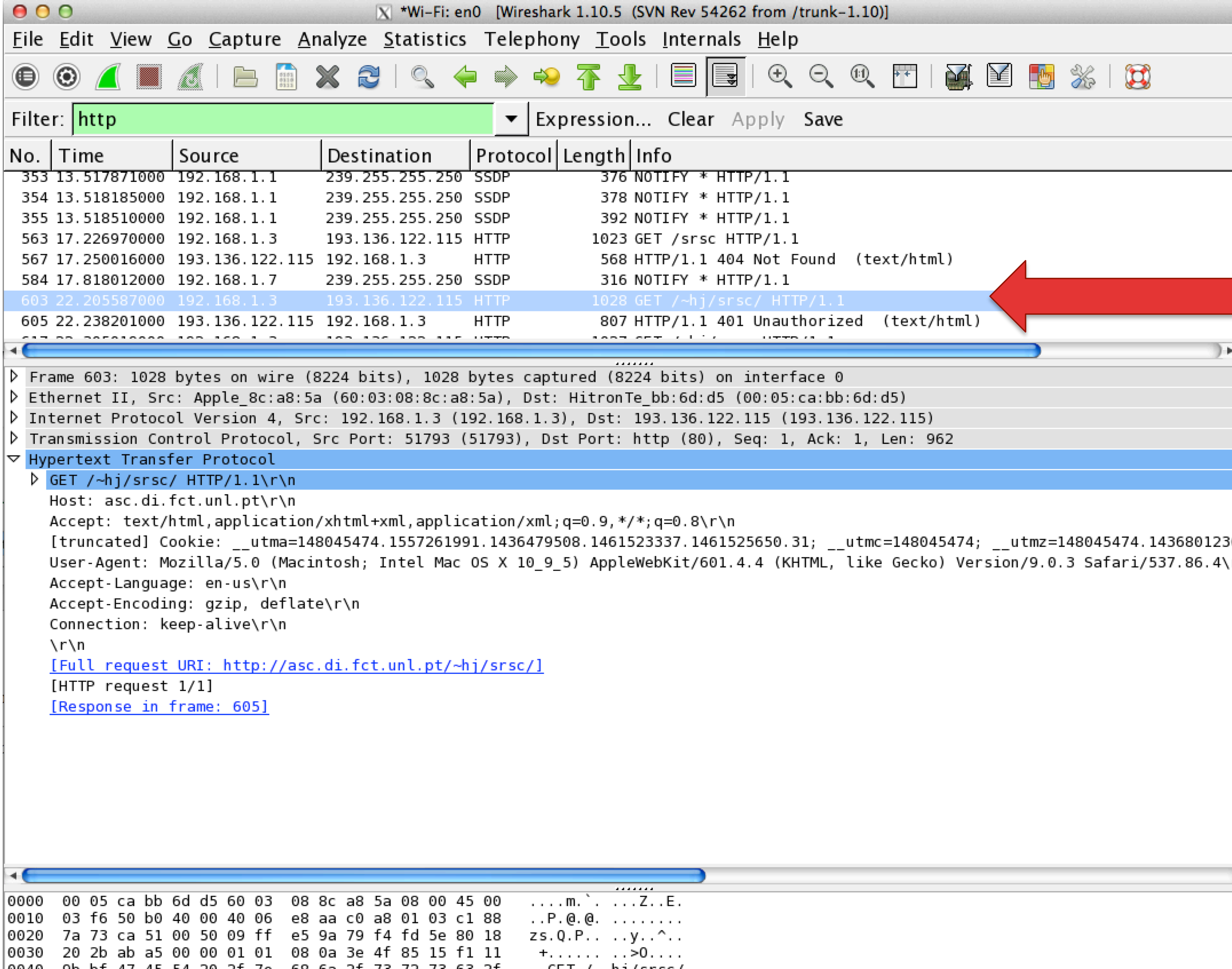
HTTP Base Athent. And Access Control

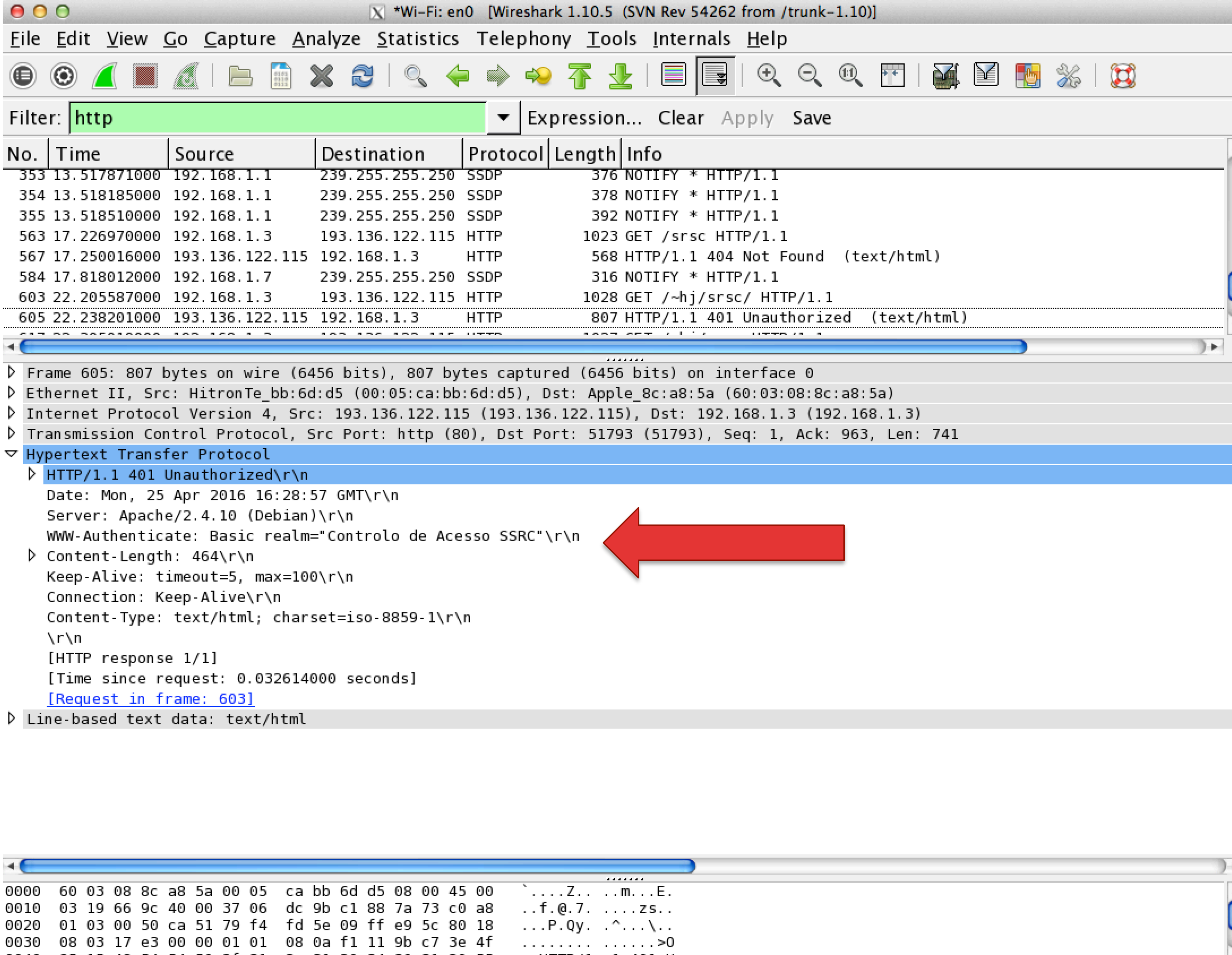
- Expression of HTTP Base-Authentication DAC policy file: the role of .htaccess in the doc-hierarchy in current implementations
- Remote HTTP access users are supposed to be created and authenticated by passwords
  - Different than OS defined users
- Use of password-based file formats


Departamento de Informática
asc.di.fct.unl.pt/~hj/srsc
iCloud Facebook Twitter Wikipedia Yahoo News Popular
FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE NOVA
Contactos WIKI Log in
Departamento de Informática
To view this page, you must log in to this area on asc.di.fct.unl.pt:80:
Controlo de Acesso SSRC
Your password will be sent unencrypted.
Name:
Password:
Remember this password in my keychain
Cancel
Log In
Apresentação
Media
Departamento
Oferta Formativa
Investigação
Pessoas
Candidato
Estudante
Ligação à Sociedade
Contactos de Suporte
Eventos
27 Abr [Seminário] Type-Based Analysis For Session Inference By Carlo Spaccasassi (Trinit...
29 Abr [Prova Académica] Provas MIEI - David Pereira Alves Neves Lopes
2 Mai [Prova Académica] Provas MIEI - Diogo João Costa Canteiro
5 Mai [Prova Académica] Provas MIEI - Bruno Filipe Gonçalves Candeias
Notícias
Daniel Casadinho recebe prémio da NOVA como melhor aluno do 1º ano do MIEI em 2014/15
A bolsa "Caloiros da NOVA" pretende premiar os melhores estudantes do 1.º ano de Licenciaturas e de Mestrados Integrados da Universidade NOVA de Lisboa.
22-04-2016
2016
TOPAS LX
TORNEIO DE PROGRAMAÇÃO PARA ALUNOS DO SECUNDÁRIO
www.di.fct.unl.pt/topas-lx-2016
INSCREVE-TE JÁ
DEPARTAMENTO DE INFORMÁTICA DA NOVA
6 MAIO
INSCRIÇÕES ATÉ 2 MAIO 2016

HTTP Base Athent. Traffic

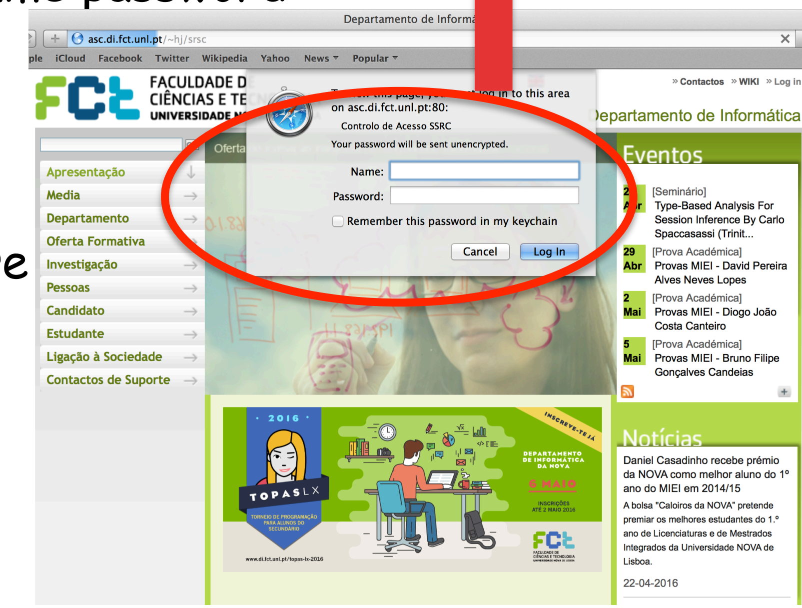
- Ex., in this case: captured by Wireshark


*Wi-Fi: en0 [Wireshark 1.10.5 (SVN Rev 54262 from /trunk-1.10)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: http
Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
353 13.517871000 192.168.1.1 239.255.255.250 SSDP 376 NOTIFY * HTTP/1.1
354 13.518185000 192.168.1.1 239.255.255.250 SSDP 378 NOTIFY * HTTP/1.1
355 13.518510000 192.168.1.1 239.255.255.250 SSDP 392 NOTIFY * HTTP/1.1
563 17.226970000 192.168.1.3 193.136.122.115 HTTP 1023 GET /srsc HTTP/1.1
567 17.250016000 193.136.122.115 192.168.1.3 HTTP 568 HTTP/1.1 404 Not Found (text/html)
584 17.818012000 192.168.1.7 239.255.255.250 SSDP 316 NOTIFY * HTTP/1.1
603 22.205587000 192.168.1.3 193.136.122.115 HTTP 1028 GET /~hj/srsc/ HTTP/1.1
605 22.238201000 193.136.122.115 192.168.1.3 HTTP 807 HTTP/1.1 401 Unauthorized (text/html)
Frame 603: 1028 bytes on wire (8224 bits), 1028 bytes captured (8224 bits) on interface 0
Ethernet II, Src: Apple_8c:a8:5a (60:03:08:8c:a8:5a), Dst: HitronTe_bb:6d:d5 (00:05:ca:bb:6d:d5)
Internet Protocol Version 4, Src: 192.168.1.3 (192.168.1.3), Dst: 193.136.122.115 (193.136.122.115)
Transmission Control Protocol, Src Port: 51793 (51793), Dst Port: http (80), Seq: 1, Ack: 1, Len: 962
Hypertext Transfer Protocol
GET /~hj/srsc/ HTTP/1.1\r\n
Host: asc.di.fct.unl.pt\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
[truncated] Cookie: __utma=148045474.1557261991.1436479508.1461523337.1461525650.31; __utmc=148045474; __utmz=148045474.1436801236
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5) AppleWebKit/601.4.4 (KHTML, like Gecko) Version/9.0.3 Safari/537.86.4\r
Accept-Language: en-us\r\n
Accept-Encoding: gzip, deflate\r\n
Connection: keep-alive\r\n
\r\n
[Full request URI: http://asc.di.fct.unl.pt/~hj/srsc/]
[HTTP request 1/1]
[Response in frame: 605]
0000 00 05 ca bb 6d d5 60 03 08 8c a8 5a 08 00 45 00 ....m.`. ...Z..E.
0010 03 f6 50 b0 40 00 40 06 e8 aa c0 a8 01 03 c1 88 ..P.@.@. ....
0020 7a 73 ca 51 00 50 09 ff e5 9a 79 f4 fd 5e 80 18 zs.Q.P.. ..y..^..
0030 20 2b ab a5 00 00 01 01 08 0a 3e 4f 85 15 f1 11 +...... ..>O....


*Wi-Fi: en0 [Wireshark 1.10.5 (SVN Rev 54262 from /trunk-1.10)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: http
Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
353 13.517871000 192.168.1.1 239.255.255.250 SSDP 376 NOTIFY * HTTP/1.1
354 13.518185000 192.168.1.1 239.255.255.250 SSDP 378 NOTIFY * HTTP/1.1
355 13.518510000 192.168.1.1 239.255.255.250 SSDP 392 NOTIFY * HTTP/1.1
563 17.226970000 192.168.1.3 193.136.122.115 HTTP 1023 GET /srsc HTTP/1.1
567 17.250016000 193.136.122.115 192.168.1.3 HTTP 568 HTTP/1.1 404 Not Found (text/html)
584 17.818012000 192.168.1.7 239.255.255.250 SSDP 316 NOTIFY * HTTP/1.1
603 22.205587000 192.168.1.3 193.136.122.115 HTTP 1028 GET /~hj/srsc/ HTTP/1.1
605 22.238201000 193.136.122.115 192.168.1.3 HTTP 807 HTTP/1.1 401 Unauthorized (text/html)
Frame 605: 807 bytes on wire (6456 bits), 807 bytes captured (6456 bits) on interface 0
Ethernet II, Src: HitronTe_bb:6d:d5 (00:05:ca:bb:6d:d5), Dst: Apple_8c:a8:5a (60:03:08:8c:a8:5a)
Internet Protocol Version 4, Src: 193.136.122.115 (193.136.122.115), Dst: 192.168.1.3 (192.168.1.3)
Transmission Control Protocol, Src Port: http (80), Dst Port: 51793 (51793), Seq: 1, Ack: 963, Len: 741
Hypertext Transfer Protocol
HTTP/1.1 401 Unauthorized\r\n
Date: Mon, 25 Apr 2016 16:28:57 GMT\r\n
Server: Apache/2.4.10 (Debian)\r\n
WWW-Authenticate: Basic realm="Controlo de Acesso SSRC"\r\n
Content-Length: 464\r\n
Keep-Alive: timeout=5, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=iso-8859-1\r\n
\r\n
[HTTP response 1/1]
[Time since request: 0.032614000 seconds]
[Request in frame: 603]
Line-based text data: text/html
0000 60 03 08 8c a8 5a 00 05 ca bb 6d d5 08 00 45 00
0010 03 19 66 9c 40 00 37 06 dc 9b c1 88 7a 73 c0 a8
0020 01 03 00 50 ca 51 79 f4 fd 5e 09 ff e9 5c 80 18
0030 08 03 17 e3 00 00 01 01 08 0a f1 11 9b c7 3e 4f

HTTP Base Athent. Protocol Summary

- Server Side (HTTP Header), Authentication Field
  - WWW-Authenticate: Basic realm="*WallyWorld*"
- Client
  - Ask user for username/password
  - Combine both in a string str= username:password
  - Compute BASE64 (str) following RFC2045-MIME
  - The authorization method and a space i.e. "Basic " is then put before encoded string
  - Repeat the REQUEST with the Authentication Field in the HEADER (GET)



Authorization: Basic c3JzYzE1MTY6dGhpc2lzbm90c29zZWNyZXRhc3NlZW1zCg==

Is it safe ?

This credential will be cached for all requests involving asc.di.fct.unl.pt/~hj/*

Base64 encoding/decoding: **not safe for this**

hj-mbp:~ hj$ **echo "srsc1516:thisisnotsosecretasseems" | base64**
c3JzYzE1MTY6dGhpc2lzbm90c29zZWNyZXRhc3NlZW1zCg==

hj-mbp:~ hj$ **echo "c3JzYzE1MTY6dGhpc2lzbm90c29zZWNyZXRhc3NlZW1zCg==" | base64 -D**
srsc1516:thisisnotsosecretasseems

It would be better ….

hj-mbp:~ hj$ **echo "srsc1516:thisisnotsosecretasseems" | openssl dgst -sha512**
2cd5b243a82a0f48c9c0d53034e3b7615e46ef408609dd0703771c65393634962421606f2eb75599f747e4b1a65a94a563580ee62d639f5fc61317406b3b8ef8

How to prevent this and how to do it better ?