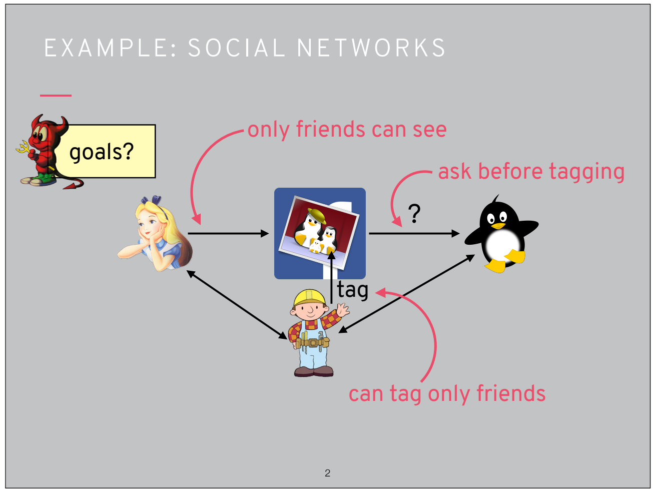
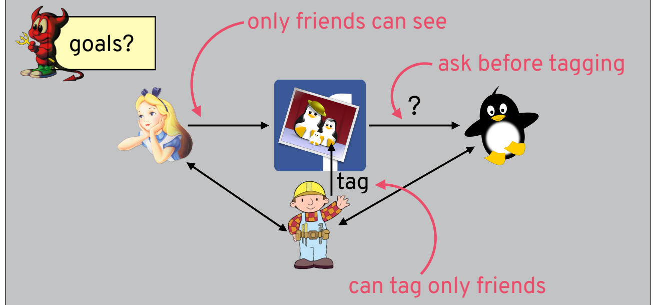
SECURITY (COMP0141): TYPES OF ACCESS CONTROL

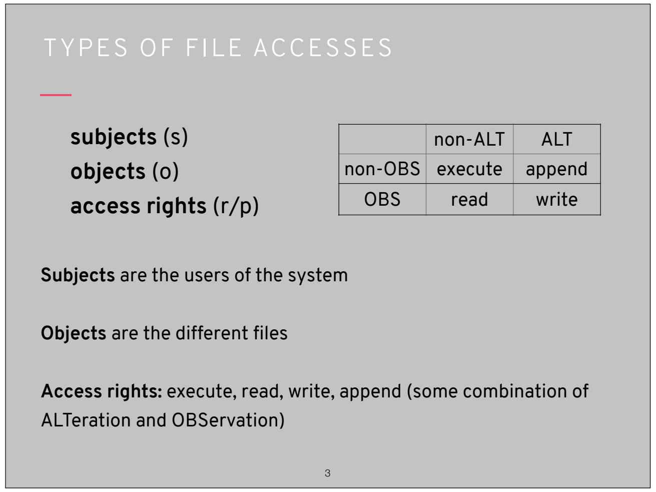
UCL



EXAMPLE: SOCIAL NETWORKS



Let’s go back and consider that example of social networks



TYPES OF FILE ACCESSES

subjects (s)
objects (o)
access rights (r/p)

	non-ALT	ALT
non-OBS	execute	append
OBS	read	write

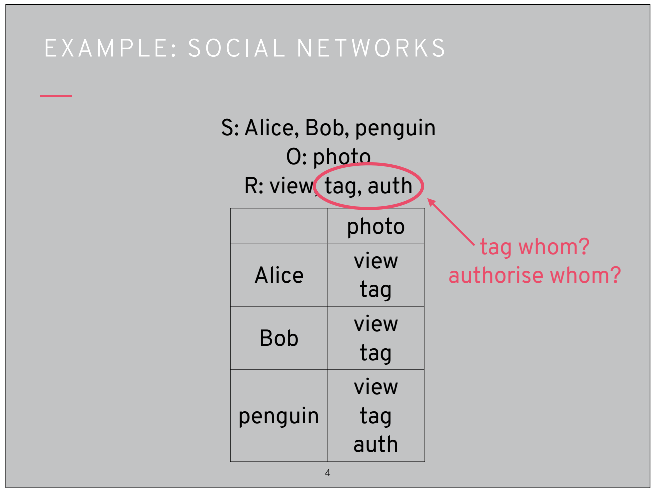
Subjects are the users of the system

Objects are the different files

Access rights: execute, read, write, append (some combination of ALTeration and OBServation)

3

And similarly recall the definition of an access control matrix



EXAMPLE: SOCIAL NETWORKS

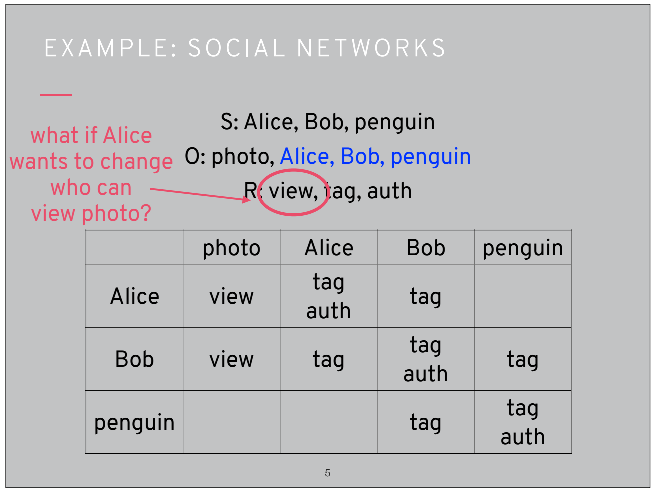
S: Alice, Bob, penguin
O: photo
R: view, tag, auth

	photo
Alice	view tag
Bob	view tag
penguin	view tag auth

tag whom?
authorise whom?

4

Access control matrices don’t immediately work as well in this setting because the permissions are more nuanced



EXAMPLE: SOCIAL NETWORKS

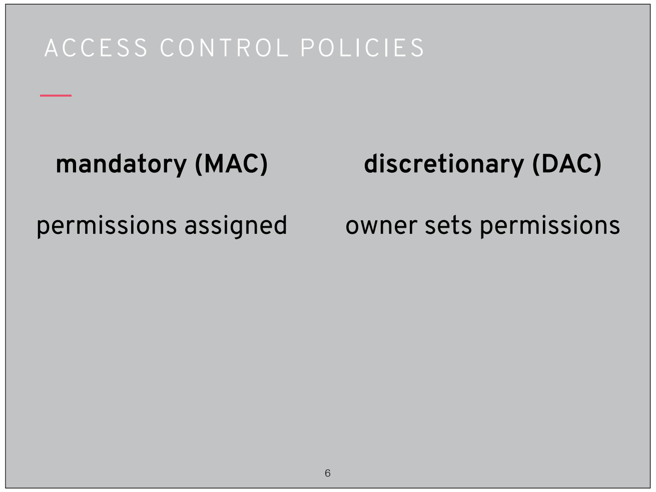
S: Alice, Bob, penguin
O: photo, Alice, Bob, penguin
R: view, tag, auth

what if Alice wants to change who can view photo?

	photo	Alice	Bob	penguin
Alice	view	tag auth	tag	
Bob	view	tag	tag auth	tag
penguin			tag	tag auth

5

To address this we also add subjects as objects and allow them to act on each other. Still though, rights might change over time (like if Alice accepts a new friend request or blocks somebody)

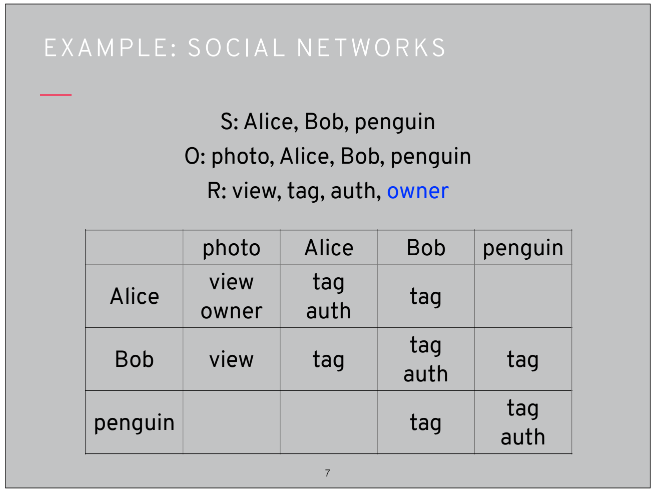


ACCESS CONTROL POLICIES

mandatory (MAC)	discretionary (DAC)
permissions assigned	owner sets permissions

6

Access control policies can be broken down into two types: mandatory (permissions are assigned and cannot be updated) or discretionary (the owner of an object gets to set their own permissions). Different ones are useful in different places: MAC is good in a large hierarchical organisation (like a hospital), and DAC is good in a more user-centric environment (like social media)



EXAMPLE: SOCIAL NETWORKS

S: Alice, Bob, penguin

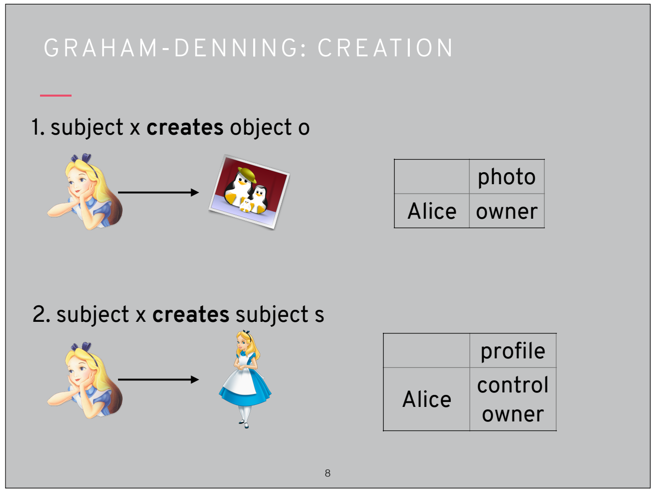
O: photo, Alice, Bob, penguin

R: view, tag, auth, owner

	photo	Alice	Bob	penguin
Alice	view owner	tag auth	tag	
Bob	view	tag	tag auth	tag
penguin			tag	tag auth

7

Another special type of right is ownership, owner should be given special privileges for the objects they create



GRAHAM-DENNING: CREATION

1. subject x **creates** object o

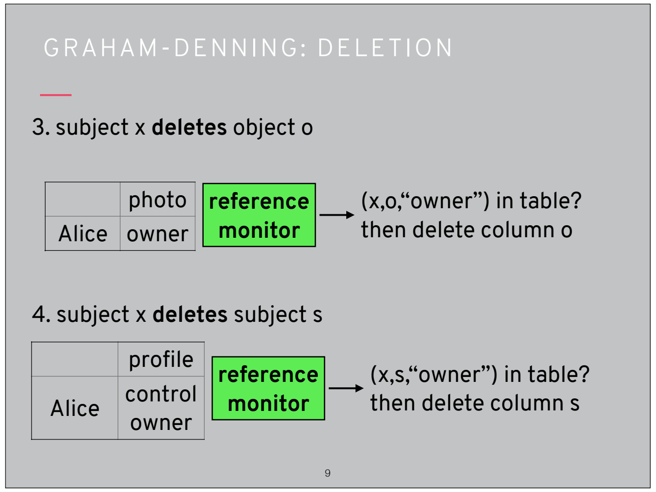
	photo
Alice	owner

2. subject x **creates** subject s

	profile
Alice	control owner

8

All of this can be captured in the Graham-Denning model (https://en.wikipedia.org/wiki/Graham-Denning_model), which lays out eight rules about access rights and how different actions get encoded into these access control matrices. 1 & 2 are about creation (posting a photo or creating an alias)



GRAHAM-DENNING: DELETION

3. subject x **deletes** object o

	photo
Alice	owner

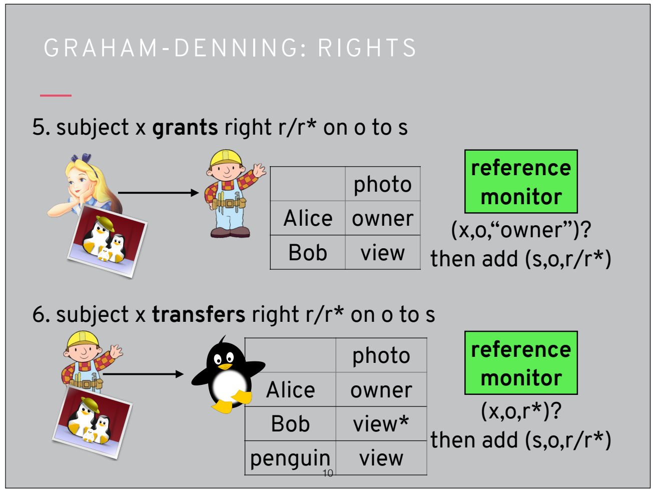
reference monitor → (x,o,“owner”) in table? then delete column o

4. subject x **deletes** subject s

	profile
Alice	control owner

reference monitor → (x,s,“owner”) in table? then delete column s

3 & 4 are about deletion (taking down a photo or deleting an alias)



GRAHAM-DENNING: RIGHTS

5. subject x **grants** right r/r* on o to s

	photo
Alice	owner
Bob	view

reference monitor

(x,o,“owner”)? then add (s,o,r/r*)

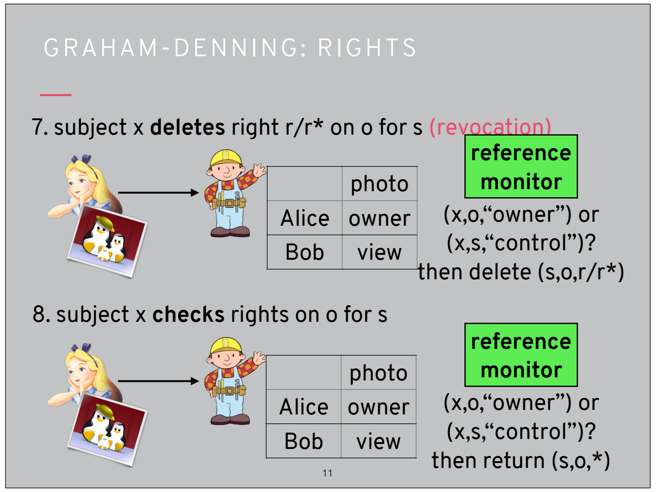
6. subject x **transfers** right r/r* on o to s

	photo
Alice	owner
Bob	view*
penguin	view

reference monitor

(x,o,r*)? then add (s,o,r/r*)

5 & 6 are about updating rights: granting rights (letting a friend see a photo) and transferring rights (letting friends control permissions of friends of friends)



GRAHAM-DENNING: RIGHTS

7. subject x **deletes** right r/r* on o for s (revocation)

	photo
Alice	owner
Bob	view

reference monitor

(x,o,“owner”) or (x,s,“control”)? then delete (s,o,r/r*)

8. subject x **checks** rights on o for s

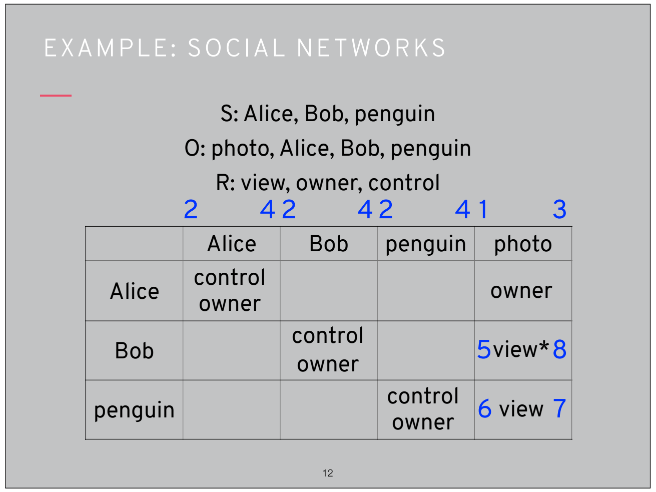
	photo
Alice	owner
Bob	view

reference monitor

(x,o,“owner”) or (x,s,“control”)? then return (s,o,*)

11

7 is about deleting rights (blocking someone) and 8 is about checking rights (who can see this photo?)



EXAMPLE: SOCIAL NETWORKS

S: Alice, Bob, penguin

O: photo, Alice, Bob, penguin

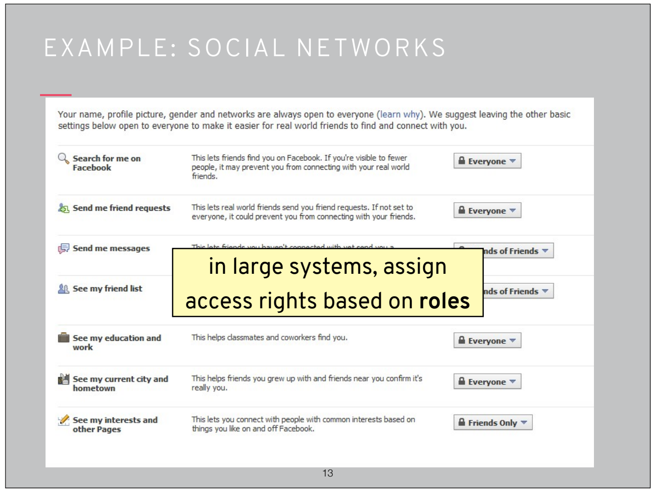
R: view, owner, control

2 4 2 4 2 4 1 3

	Alice	Bob	penguin	photo
Alice	control owner			owner
Bob		control owner		5view*8
penguin			control owner	6 view 7

12

Can put it all together to see how matrix gets built and updated



EXAMPLE: SOCIAL NETWORKS

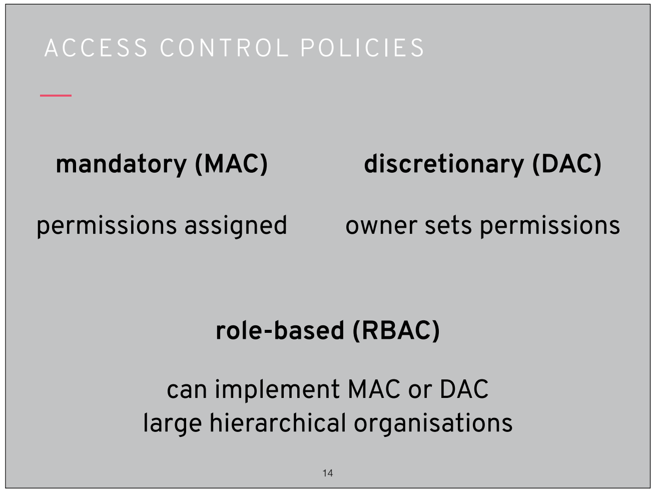
Your name, profile picture, gender and networks are always open to everyone (learn why). We suggest leaving the other basic settings below open to everyone to make it easier for real world friends to find and connect with you.

Search for me on Facebook	This lets friends find you on Facebook. If you're visible to fewer people, it may prevent you from connecting with your real world friends.	Everyone
Send me friend requests	This lets real world friends send you friend requests. If not set to everyone, it could prevent you from connecting with your friends.	Everyone
Send me messages	This lets friends you haven't connected with yet send you a	nds of Friends
See my friend list		nds of Friends
See my education and work	This helps classmates and coworkers find you.	Everyone
See my current city and hometown	This helps friends you grew up with and friends near you confirm it's really you.	Everyone
See my interests and other Pages	This lets you connect with people with common interests based on things you like on and off Facebook.	Friends Only

in large systems, assign access rights based on **roles**

13

But of course we don't do things at this level of detail in practice! We really assign access rights based on roles, like 'friends' or 'friends of friends'. This is called role-based access control (RBAC)



ACCESS CONTROL POLICIES

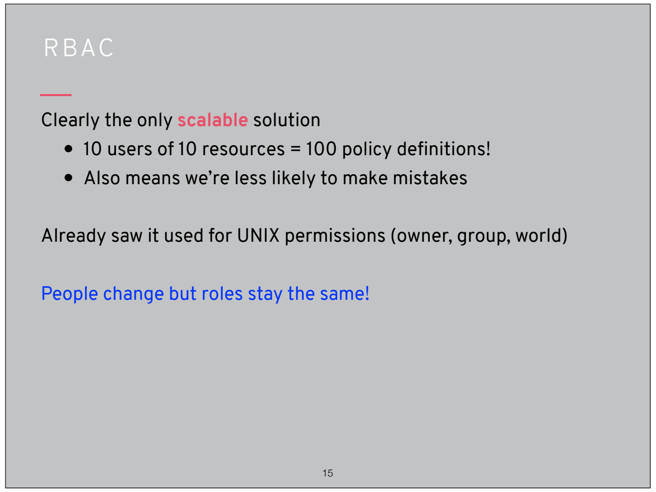
mandatory (MAC)	**discretionary (DAC)**
permissions assigned	owner sets permissions

role-based (RBAC)

can implement MAC or DAC
large hierarchical organisations

14

RBAC can implement either type of access control



RBAC

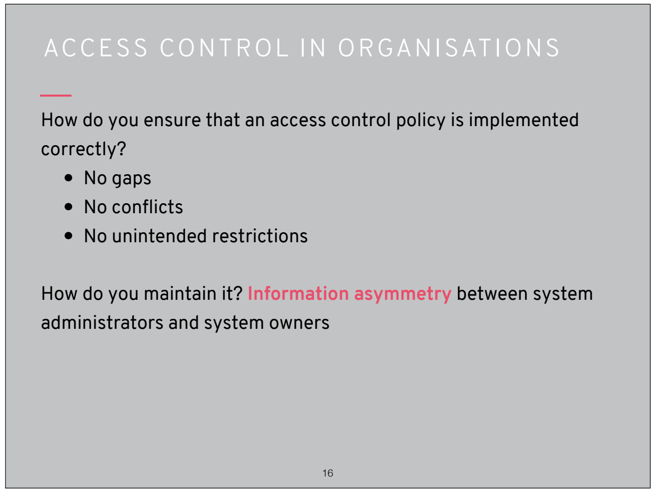
Clearly the only scalable solution

- 10 users of 10 resources = 100 policy definitions!
- Also means we’re less likely to make mistakes

Already saw it used for UNIX permissions (owner, group, world)

People change but roles stay the same!

15



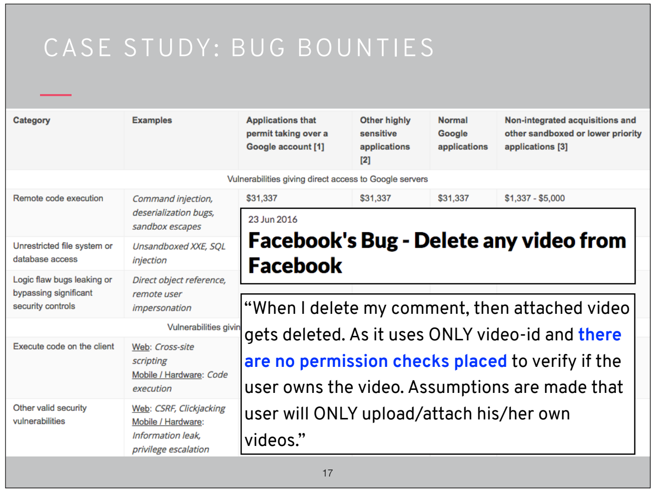
ACCESS CONTROL IN ORGANISATIONS

How do you ensure that an access control policy is implemented correctly?

- No gaps
- No conflicts
- No unintended restrictions

How do you maintain it? Information asymmetry between system administrators and system owners

16



CASE STUDY: BUG BOUNTIES

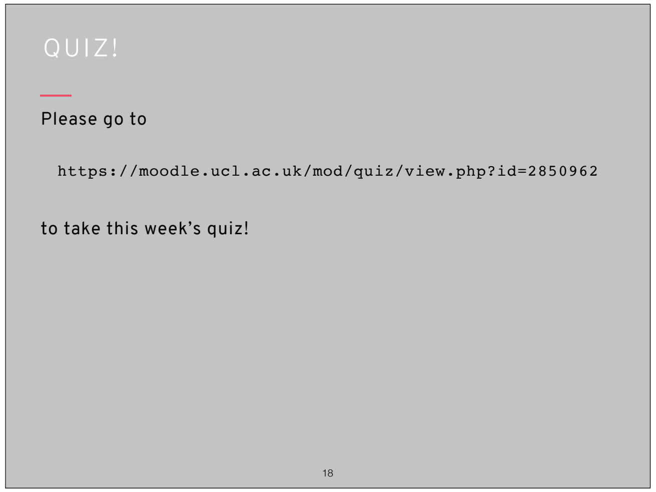
Category	Examples	Applications that permit taking over a Google account [1]	Other highly sensitive applications [2]	Normal Google applications	Non-integrated acquisitions and other sandboxed or lower priority applications [3]
Vulnerabilities giving direct access to Google servers					
Remote code execution	*Command injection, deserialization bugs, sandbox escapes*	$31,337	$31,337	$31,337	$1,337 - $5,000
Unrestricted file system or database access	*Unsandboxed XXE, SQL injection*				
Logic flaw bugs leaking or bypassing significant security controls	*Direct object reference, remote user impersonation*				
Vulnerabilities givin					
Execute code on the client	Web: *Cross-site scripting* Mobile / Hardware: *Code execution*				
Other valid security vulnerabilities	Web: *CSRF, Clickjacking* Mobile / Hardware: *Information leak, privilege escalation*				

23 Jun 2016

Facebook's Bug - Delete any video from Facebook

"When I delete my comment, then attached video gets deleted. As it uses ONLY video-id and **there are no permission checks placed** to verify if the user owns the video. Assumptions are made that user will ONLY upload/attach his/her own videos."

Bug bounty programs illustrate that large organisations always have gaps in access control policies, this is ultimately why access control is so important (because it's so hard to get right!). The specific Facebook example can be found at https://pranavhivarekar.in/2016/06/23/facebooks-bug-delete-any-video-from-facebook/



QUIZ!

Please go to

```
https://moodle.ucl.ac.uk/mod/quiz/view.php?id=2850962
```

to take this week's quiz!