STUDENT HANDOUT · WEEK 13

Identity, Access, and Automation

SY0-701 objectives 4.6, 4.7. Notes, exercises, review questions, and the lab.

Name	
Date	

THIS WEEK YOU WILL

1. Describe the account lifecycle from identity proofing and provisioning to attestation and removal.
2. Explain how federation and SSO work with LDAP, SAML, and OAuth, and why interoperability matters.
3. Choose among mandatory, discretionary, role-based, rule-based, and attribute-based access control.
4. Design MFA with factors from different categories and compare tokens, biometrics, and security keys.
5. Apply password best practices, password managers, and passwordless sign-in to a scenario.
6. Explain just-in-time permissions, password vaulting, and ephemeral credentials for privileged access.
7. Weigh the benefits of security automation against its complexity, cost, and supportability.

KEY IDEAS

An account's life runs from proofing to removal

1. Proof identity	Verify the person is who they claim to be
2. Provision	Create the account from a role template
3. Assign	Grant only the permissions the job needs
4. Attest	Managers confirm access is still needed
5. Deprovision	Disable on the last day; remove access everywhere

One login, many services, through trust

Protocol	What it does	Typical use
LDAP	Queries and updates a directory of users	Directory lookups; LDAPS on TCP 636
SAML	Signed XML assertions from IdP to SP	Web SSO to business SaaS apps
OAuth	Grants an app limited access with a token	Let an app read your calendar
OpenID Connect	Adds authentication on top of OAuth	Sign in with an existing account
Federation	Trust between separate identity domains	Partner staff use their own logins

Who decides access, and on what basis

Model	Who or what decides	Example
Mandatory (MAC)	System labels and clearances	Secret clearance reads Secret files
Discretionary (DAC)	The owner of the resource	A file owner shares a folder
Role-based (RBAC)	The user's job role	The Nurses group can open charts
Rule-based	Admin-set rules applied to everyone	Firewall ACL; no sign-in from abroad
Attribute-based (ABAC)	Attributes of user, resource, context	Doctor, own patient, on clinic network
Time-of-day restrictions	The clock	Front desk sign-in 7 a.m. to 7 p.m. only

MFA means factors from different categories

Something you know	A password, PIN, or answer to a security question (Weakest alone; can be phished)
Something you have	A hard token, authenticator app, smart card, or security key (TOTP app, FIDO2 key)
Something you are	Biometrics: fingerprint, face, iris, or voice (Tuned by FAR, FRR, and CER)
Somewhere you are	Location from GPS, IP address, or network (Block sign-ins from abroad)

Long, unique, and stored in a manager

Password best practices	Beyond the password

Length: long passphrases matter most	Password managers make unique passwords practical
Complexity: mixed character types if policy requires	Passwordless: FIDO2 passkeys, Windows Hello
Reuse: password history blocks old passwords	The private key never leaves the device
Expiration: maximum age forces a change	Resists phishing because the site is checked
Age: minimum age stops cycling back	

Give admin rights only when and while needed

Just-in-time	Admin rights granted for a task, then removed automatically (Two hours, approved by ticket)
Password vaulting	Admin passwords kept in a vault, checked out, and rotated (Session recorded on checkout)
Ephemeral credentials	Short-lived credentials created for one use, then expired (Cloud token valid for one hour)

Automate the tasks people repeat or forget

Use case	Example
User and resource setup	New hire gets an account; templates build servers
Guard rails	Block a public storage bucket before it deploys
Security groups	Add users to groups by role; remove on job change
Tickets and escalation	Alert opens a ticket; pages on-call if unanswered
Services and access	Disable an account and its sessions at termination
CI, testing, and APIs	Scan every build; connect tools through their APIs

Automation multiplies whatever it is given

Benefits	Other considerations
Efficiency, time saving, and faster reaction time	Complexity: harder to understand and debug
Enforces baselines and standard configurations	Cost to build, license, and maintain
Scales without adding staff	Single point of failure if the engine breaks
Workforce multiplier for a small team	Technical debt from quick, undocumented scripts
Employee retention: less tedious work	Ongoing supportability when the author leaves

EXERCISE 1: EXERCISE 1: NAME THE ACCESS CONTROL MODEL

6 MINUTES · INDIVIDUAL

Name the access control model each scenario describes. For the last two, name the principle broken and the fix.

#	Prompt	Your answer
1	Files are labeled Top Secret and checked against clearance	
2	Anyone in the Billing group can open the billing system	
3	A file's owner decides to share it with a coworker	
4	Access only for doctors, to their own patients, from the clinic	
5	A firewall blocks all inbound traffic except TCP 443	
6	Temporary staff accounts cannot sign in on weekends	
7	A new hire's account is copied from a 10-year employee's	
8	The IT intern is a domain admin so tickets close faster	

EXERCISE 2: EXERCISE 2: AUTOMATE THE CLINIC'S IAM WORK

8 MINUTES · SMALL GROUPS

For each task, decide whether to automate it, name the use case, and give one benefit and one risk.

#	Prompt	Your answer
1	Create accounts and groups for each new hire from the HR system	
2	Disable accounts when HR marks a termination	
3	Weekly report of accounts with no MFA or no login in 30 days	
4	Approve a doctor's request to see another clinic's records	
5	Isolate a laptop automatically when EDR detects ransomware	
6	Rotate the domain admin password after each use	

REVIEW QUESTIONS

1. What is identity proofing, and when does it happen?

2. Why is OAuth not, by itself, an authentication protocol?

3. How does role-based access control differ from rule-based access control?

4. Is a password plus a security question multifactor authentication? Why or why not?

5. What is the difference between password vaulting and ephemeral credentials?

KEY TAKEAWAYS

- **Access follows the person.** Proof, provision, review, and remove access the day people leave.
- **Know who decides access.** MAC, DAC, RBAC, rule-based, and ABAC each answer who decides.
- **Factors must differ in kind.** Two passwords are one factor; FIDO2 security keys resist phishing.
- **Automate, then support it.** Scripts act fast and consistently, and must be documented and owned.

LAB 13: MFA, LEAST PRIVILEGE, AND A SHORT AUTOMATION SCRIPT

60 MINUTES

Use your Ubuntu lab VM, the Windows VM if you have it, and any authenticator app on your phone. Take a snapshot first and keep a second terminal session open while you change SSH and PAM settings, because a mistake can lock you out. Work only in your own lab.

Steps

1. Take a snapshot of the Ubuntu VM named pre-lab11.
2. Apply least privilege. Create two users with sudo adduser frontdesk and sudo adduser itadmin, a group with sudo groupadd billing, then run sudo usermod -aG billing frontdesk and sudo usermod -aG sudo itadmin. Confirm with id frontdesk and sudo -l -U frontdesk.
3. Restrict a folder to its group: sudo mkdir /srv/billing, sudo chgrp billing /srv/billing, and sudo chmod 770 /srv/billing. Show that itadmin can reach it only by using sudo and frontdesk can reach it directly.
4. Add MFA to SSH. Run sudo apt install libpam-google-authenticator, then sign in as itadmin and run google-authenticator, scanning the QR code with your app. Add the line auth required pam_google_authenticator.so nullok to /etc/pam.d/sshd.
5. In /etc/ssh/sshd_config, set KbdInteractiveAuthentication yes (older releases call it ChallengeResponseAuthentication), run sudo systemctl restart ssh, and test ssh itadmin@localhost. It should ask for the password and a verification code. The nullok option lets users without MFA still sign in, which your script will catch.
6. Write mfa-report.sh. Loop over accounts with a UID of 1000 or higher (awk -F: '$3>=1000 && $3<65534 {print $1}' /etc/passwd), print NO MFA for each user whose home folder has no .google_authenticator file, then list accounts with no login in 30 days with lastlog -b 30. Run it with bash mfa-report.sh.
7. Optional on Windows: in PowerShell, run Get-LocalUser | Where-Object { $_.Enabled -and $_.LastLogon -lt (Get-Date).AddDays(-30) } | Select-Object Name, LastLogon to list enabled accounts with no logon in 30 days. Accounts that never logged on also appear.
8. Act on the report: set up MFA for one flagged account and lock a stale one with sudo usermod -L USERNAME (or Disable-LocalUser -Name USERNAME on Windows). Rerun the script and confirm the findings changed.

Hand in

A short report with screenshots of the group and permission checks, the SSH login asking for a verification code, your script's source code, and its output before and after you acted on it.

Questions to answer in your report

1. Which least privilege problem did your group and folder setup prevent, and how would you prove it to an auditor?

2. What does the nullok option allow, and why would you remove it once every user has MFA set up?
3. How would you run your script automatically every week, and where should its results go so someone acts on them?
4. Name one automation risk from objective 4.7 that applies to your script, and how you would reduce it.