**BOOK-1: Audit in the enterprise and cloud**

Measure and report on **risk** | Verify compliance with policies/laws/regulations.

*Auditor Roles*: Mgmt, Compliance, Operations, Security

*Audit techniques*: Start small; risk-driven; outside in; bite-sized pieces; Findings must be relevant;

**TERMS***: Audit*: Test of an assertion (how do you know..) | *Assessment*: Are there opportunities for improvement? | *Scope*: What are we looking at? | *Objective*: Why are we looking?

**Primary objective**: Can be met by measuring & reporting on how well a system/process measure up to best practice or policy.

Secondary objective: Influence others to reduce risk; raising awareness.

**Auditor qualities**: *Independence/objectivity* [Segregation of audit/consulting practices required in US under SOX Act]; *Competence/proficiency* [knowledge of systems, controls, technologies; SME can help; Continuing education is must]; *Due professional care* [During planning, execution, reporting; opposite of negligence]; *Professional skepticism* [protects against risk of misunderstandings & material misstatements; auditors should be skeptical but friendly.]

**Internal, External, Advisory Auditors**: Internal work for orgs but independent; External work for regulators or firms; Auditors can provide advisory functions (offer opinions on new control designs)

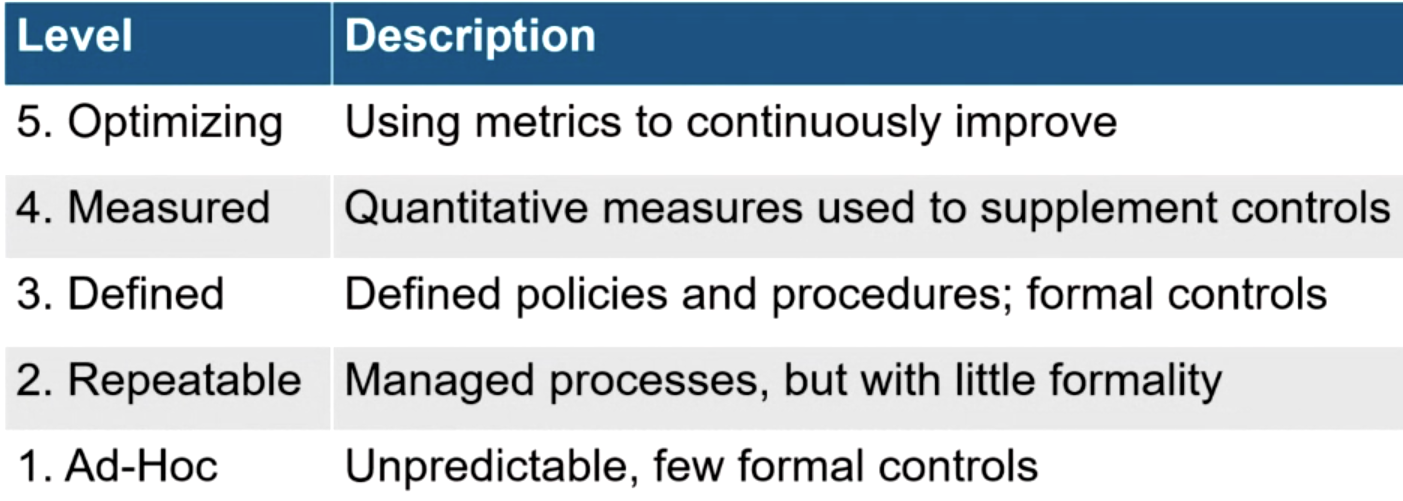
**Policy** -> answers who, what, why | **Procedures** -> Who does what, when and how | Audits measure performance of org on policy and procedure; **Incident handling and auditing** serve as policy/procedure assessment **tools**.

**“Speed Limit” Test**: Policies and Standards are not meaningful without **measures** for controlling risk.

Communicating risks: Cooperative & enable remediation

**Audit Process**:

* Audit Planning -> Entrance conference (1st meet w/ client) -> Fieldwork -> Exit Conference -> Reporting
  + Planning or Pre-audit activities like
    - Research [corp policy, industry best practices, frameworks & benchmarks],
      * Understanding org maturity [CMM]]:



* + - Scope: Agreed b/w management and audit team; define 1st time out; don’t back more than once
      * Scope can be difficult in cloud: permissions to test; verifying ownership of assets, etc.
    - Audit strategy: WHAT and HOW to check;
    - Checklist: **Work Program Items**: Statement of purpose; Best practices;
      * *References*: prefer internet; Avoid auditor name; Communicate checklist; discuss objections
      * Sources of guidance: CIS, ISACA, DISA, AICPA, IIA, OWASP, FFIEC [questions to determine what board does like delegating monitoring task to steering committee, getting reports]
    - CIS Internetwork Operating Systems (**IOS**) 15 benchmark
      * AUDIT: #show run | incl aaa authentication login

RECOMMENDATION: #aaa authentication login {default | aaa\_list\_name} [passwd-expiry] method1 [method2]

* + - Audit procedures
  + Importance of **inventory** to the auditor: Network maps (ask for diagrams, never trust those directly, validate with scan tools like nmap)
    - **Nmap host discovery**: for local hosts (ARP ping) | for remote hosts (ICMP ping, TCP SYN to 443, TCP ACK to 80) | use -PE flag (ICMP echo), use -PS flag (TCP SYN), use -PA flag (TCP ACK)
    - **TCP Scans**: Stealth (-sS): no full 3-way handshake (SYN, SYN/Ack, RST) | Full Connect (-ST): full 3-way handshake | ACK scan (-sA): TCP ACK flag; won’t pass firewall
    - **UDP Scans**: -sU | very slow

#nmap -sU -sT -pU:53,111,T:22-25,80,443,53 10.50.7.30

* + - Service versions: -sV | Detect OS version: -O

#nmap -sV -sT -p 22 10.50.7.20

* + - NMAP output formats: -oN (normal text) | -oX (XML) | -oG (greppable) | -oA (all)
    - NSE scripts: -sC or –script=default : Runs all scripts

#nmap -p443 10.50.7.20,50 –script ssl-cert,ssl-enum-ciphers

Vulnerability scanning market: Qualys or Tenable

**Nessus Essential**: Scan types: Host discovery, Basic network (**used by auditors**), Credentialed patch, Advanced scan (by default no plugins), Malware & vuln detection scans

**Nessus Compliance Scans**: Paid | CIS benchmarks, DISA STIGs, PCI-DSS

**Nessus plugins**: define tests to run against targets | can write own plugins

Dashboards: Qualys

**Risk**

Threat x vulnerability = Risk

**Traditional Risk Assessment**: Overly subjective; Difficult for management to use for decision-making

**Cause-Consequence Analysis (CCA)**: RCA; Designing controls; Find consequences of failures using event trees or fault trees

*Event Tree*: Analyze *consequences* of adverse event or control failure | helps when a failure is critical to organization

Technique: Identify adverse events -> Existing controls (Preventive, Detective, Corrective) -> Effects of control failure

Availability (e.g. power failure), Integrity (e.g database server)

*Fault Tree*: Analyze *underlying causes* of critical failure event: Q: For this to occur, what underlying cause must be true?

Uses AND or OR operators for connected events | Helps in Root-cause analysis (RCA)

**Time-Based Security (TBS)**: Time as measure to know we have ‘enough’ security as time is money

Measure: How much security is too much? | sufficiency of security controls | how much more is needed in numbers

Like qualitative risk assessment | must be reproducible

*TBS with Defense-in-depth*: P > D + R => this means time of Protect > time of (Detect + Respond)

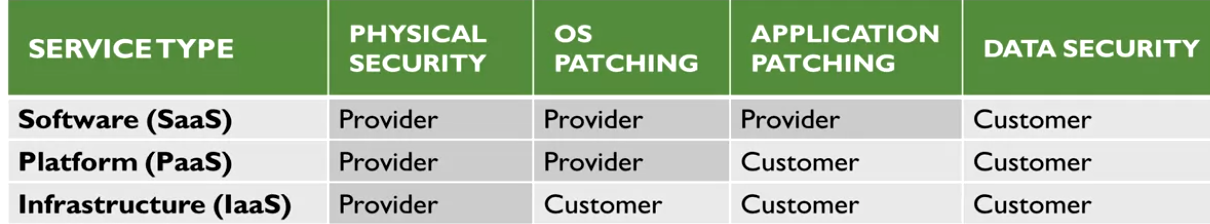
Based on that, we can computer **cost/benefit analysis (CBA)**

One can audit : Detection time or Response time

**Risk control concepts:**

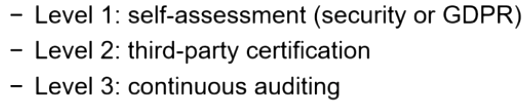
Separation of duties | Principles of least privilege | Economy of mechanisms (simpler design) | Complete mediation (one system like SSO for everyone)

Audit cloud strategies: Private, Public, Hybrid cloud | SaaS, PaaS, IaaS | Shared responsibility model



Security **OF** the cloud: CSA **CCM** (Cloud Control Matrix) and **CAIQ** (Consensus Assessment Imitative Questionnaire)

**CSA STAR** (Security, Trust Assurance and Risk) program: Certification program.



On Privacy: STAR level-1: GDPR CoC Self-Assessment and STAR level-2: CoC certification

CSA STAR attestation – SOC2 + CCM | STAR certification: ISO27001 + CCM

**CSA STAR registry**: Search for your cloud provider’s STAR level | can download PDF report of public version of those clouds

**Audit considerations: SaaS**

Audit compliance reports | proper user configuration | Audit **SaaS lifecycle** (Evaluation, Adoption, Usage, Termination)

**Audit considerations: PaaS**

Real-time protection (Firewalls, Monitoring) | 3rd party & custom apps, data | dev process of apps | Logs & monitoring

**Tagging**: name=value; AWS: Use tag editor on web console | Azure: Resource graph explorer | GCP: Asset Inventory tool (35-day history) | **All CSP providers use CloudQuery.io to gather inventory**.

Cloud **Network Resources**: AWS [Virtual Private Cloud, VPC] | Azure [Virtual Networks] | GCP [Virtual Private Cloud]

Cloud **Compute Resources**: AWS [EC2] | Azure [VM] | GCP [Compute Engine VM]

Cloud **Storage Resources**: AWS [S3] | Azure [Azure Storage] | GCP [Storage buckets]

Cloud **Availability & Redundancy**: AWS & Azure [regions w/ availability zones] | GCP [regions w/ zones]

Cloud **Management Interfaces**: Web portals, CLIs, APIs by all three CSPs

**Audit Entrance Meeting**: Attendees [Managers, Network or sys admins, Developers, Database admins] | Agenda [Introductions, Audit staff takes lead, Establish credibility, explain score & objectives, introduce audit process, describe test/verify procedure, establish point of contact, begin scheduling interviews & other meeting, Establish work arrangements like space to work, network, physical access, working hours, documentation needs]

**Audit Fieldwork**: Perform & document tests outlined; research into technology; document org controls; testing the controls; gather evidence | **Audit interim/Status reporting**

**Audit Exit Conference**: Final clarification of outstanding issues, facts, notes, next steps, Exit conference attendees [mgmt., auditors, those involved in audit], Agenda: Summary, Risks, Findings or concerns (pushback normal), Accept input from auditee on wording; allow staff to make any arguments for revision or severity

**Audit Reporting**: Understand consumers of that report | primary and secondary users of report

Report contents: Cover letter, executive overview (< 1 page), recipients, scope, objective, period, statement of mgmt. resp, description of measured criteria & tests, findings/observations, recommendations, limitations, auditors’ opinion

ISACA stds and guidelines are applicable for audits | ITAF (IT Audit Framework) is guide

* ISACA standard 1401 -> need a report for each audit
* ISACA standard 2401 -> required report sections and topics like subsequent events

*Board Presentation*: Executive summary | *Management presentation*: about 1 hour, full report to right people | *Technical presentation*: Executive summary, full report, take breaks, more detailed recommendations

**BOOK-2: PowerShell, Windows System, Domain Auditing**

**General OS checklist**: Demographic info, Services (N/w, Local), Users/Groups/Passwords, Protecting data, OS & App security hardening (Registry settings, Local user rights), Auditing and Logging information

**PowerShell**: Not case-sensitive, Cross-platform, Object oriented, .NET (Core) integration, Windows-native

Windows PowerShell (blue background, Win native, Full .NET framework, stuck in 5.1 version, no new features/updates) vs PowerShell Core (black background, Pwsh.exe on windows, Cross-platform, .NET core, version 7+ LTS, new features added)

Commands returns objects, not just text! Objects have properties (data) and methods (functions)

**Pipeline** (|) send output f/ one cmd to another; Allows efficient data processing; allows for inserting line breaks like grave (`)

**Command types**: Cmdlets [pre-compiled fully functional] and Functions [written in PowerShell, not pre-compiled];

**Alias** [Short names; only for command prompt use; use full names in scripts]

**Format**: “Verb-Noun” ; Noun is always singular; **Verb**: Add, Clear, Close, Copy, Enter, Exit, and so on.

**Parameters**: Modify command behavior; multiple parameters possible; In *positional* params, param names can be omitted.

**Quotation Marks**: *single quote* (literal string); *double-quotes* (variables to dynamically update); Escaped chars: for spaces

**Help**: *Get-Help Get-ADUser -Online* ; *Online* parameters helps to get most current information

**Update-Help**: Download from Microsoft website latest help info; **Save-Help**: saves help files for offline updating

For Audit Evidence Acquisition: **Selecting & sorting data** [Get-Member, Select-Object, Where-Object, Sort-Object, Get-Unique, Group-Object, PS comparison (-eq, -ne, -gt, -ge, -lt, -le) (-like, -notlike uses wildcard \*) (-match, -notmatch for regex) (-in, -notin for collection/sets) (-contains, -notcontains) & formatting cmds], **Managing output** [Format-List, Format-table, Measure-Object], **Data formats** [JSON, XML, CSV] (ConvertFrom-Json, ConvertTo-Json, ConvertTo-Xml, Select-Xml, *Export-CliXml* (can save encrypted creds on Windows using Data Protection API), Import-CliXml, ConvertFrom-CSV, ConvertTo-CSV, Import-CSV, Export-CSV), **PS scripting** [Start-Transcript, Stop-Transcript], Parameters/Functions, Looping, Conditionals

**PowerShell Scripting**: *Script*: collection of commands saved in a file to execute at will | Code reusable | Automation | Scaling

*Parameters*: Allow to change the behavior of script/function; Must be defined in the 1st executable line of the script.

*Functions*: Reusable code with name attached; ‘return’ keyword will exit function and return value; when no return, last cmd run is returned; Functions must be declared above any calls to the function (top-down parsing).

*For Loop*: Uses starting state, condition for repeating; command to execute on each repeat;

for($x=20; $x -lt 30; $x++) {

Test-NetConnection -ComputerName “10.50.7.$x”

}

*Foreach or Foreach-Object*: Loop through all objects in a collection; When using in a pipeline, both means the same.

*Conditionals (If/Else statements)*: block runs only when condition is true; true and false are built-in variables in PowerShell.

If(Test-Path -Path $filename -Type Leaf) {

“Checking scan results”

} else {

“Skipping scan check – file missing”

}

**When PowerShell is NOT available**: Use “Belt and suspenders” approach | CLI tools, GUI tools, etc.

**Windows Management Instrumentation (WMI)**: Introduced in Win NT; Related to **WBEM** (Web Based Enterprise Mgmt), **CIMv2** (Computer Information Model, by DMTF, standards-based model for info about systems)

* WMI exposes any setting for the purpose of measuring and scripting like NIC config, Desktop settings, Pwd lockout status, System config info, Event logs, etc. (**Get-CimClass**)

**Get-WMIObject** (old, deprecated, DCOM) vs. **Get-CimInstance** (New, WinRM based on SOAP, In PS Core)

When auditing Windows, start with System ‘**Demographics**’: OS type, version, system info like uptime, registered user/company, basic hardware like CPU, memory, disk, NTFS, and purpose of the host.

CLI: systeminfo | GUI: msinfo32

OS version: PS> Get-CimInstance Win32\_OperatingSystem | select Caption, Version, BuildNumber

**Get-PhysicalDisk, Get-Partition (NTFS/FAT32), Get-Volume** (filesystem)

Get-ItemProperty “HKLM:\SYSTEM\CurrentControlSet\Control\Lsa”

LimitBlankPasswordUse : 1 (1 = limits blank password, so enabled)

Auditors must check for System Patches / Updates

**Windows Patches**:

* Builds (security+non-security, new features, new version of Windows, replaced service-packs)
* Hotfixes or critical updates (Fix for single critical issue on security/stability)
  + Automatic updates, Windows update website, patch mgmt. software, manual install
  + Registry entry shown
  + Uninstall directory written to %windir% and needs reboot of Windows
* QFE fixes (Interim fix for single/specific issue, only from MS)

In case of **96 hour** to patch schedule: 24 hrs for ‘power users’ | 48 hrs for noncritical servers | 72 hrs for all users | 96 hrs for all servers.

PS>Get-Hotfix | Format-Table HotFixID, Description, InstalledBy, InstalledOn

**Patch age**: # of days since last patch was applied

PS>$lastPatchDate = (Get-Hotfix | Sort-Object InstalledOn -Descending | Select-Object -First 1).InstalledOn

**Patch velocity**: # of patches applied per day (frequency)

PS>Get-Hotfix | Group-Object InstalledOn

*Difficulties with patch validation*: Different utilities give different results | Depends on how patch level is checked (registry entry, file version/checksum, both?, manual check, etc.)

Microsoft WSUS, SCCM/MECM, PatchLink (Ivanti), Shavlik (Ivanti), IBM BigFix

*Best way to check patch*: if central tool to patch exist, query it | Authenticated vuln scans can be helpful

Nessus plugin #38153 -> Report of missing Microsoft patches

Nessus plugin #66334 -> report of missing patches for OS and installed software

**Auditing Installed Software on Windows**: PS>Get-CimInstance Win32\_Product | Select Name, Version, InstalledDate

*Osquery* can be used as well.

**Audit interview questions**: Change control policy | Scheduled maintenance | Compliance policy | Exception policy

Network and Local Services: Increases remote attack surface | Audit tools: Services: Get-Service, Get-Process, sc, tasklist, psservice | **Ports**: **Get-NetTCPConnection**, nmap, netstat

Unneeded services: like IIS, SMTP, Messenger, etc. | May contain vuln | Unused services are not liked patched | Rogue services may contain malware infection

**How to check services**? Outside (port scanner) or Inside (from host itself) | Do both !

Services mmc | psservices.exe [from Microsoft, old sysinternals/Winternals], sc.exe, tlist/tasklist |

PS>**Get-Service | Select Name, Status, StartType**

PS>**Get-Process | Select Id, Name, WorkingSet, CommandLine**

Tasklist: command-line list of running processes | allows remote queries

PS> tasklist /s 507dc

Audit toolkit: **Osquery**: Facebook 2014 | Emulates SQL database for low-level OS info like processes, kernel modules, users, applications, etc. | Works on Windows, Linux, BSD, and macOS

* Comes with SIEM and Endpoint protection tools | 270+ tables | 50+ work across all platforms

**Osqueryi** : Interactive SQL console | Run ad-hoc queries on system like osquery> select name, version, build from os\_version;

**Osqueryd**: Accepts remote requests, Runs a service, Schedule to run across enterprise, can log query to HTTPs server

Osquery>select name,version from programs where name like ‘%firefox%’;

Tables:

* Programs: List of installed programs, version number, install date
* Patches: list of all hotfixes on system
* Services: Info about all services on system

Cross-platform tables: os\_version | users, groups, users\_groups | listening\_ports | File (access to host filesystem)

Tbales included for AWS, and Azure.

* Azure\_Instance\_Tags | Azure\_Instance\_Metadata
* EC2\_Instance\_Tags | EC2\_Instance\_Metadata

Event-driven tables can log system events: PowerShell\_Events | Process\_Events

**Auditing users on Windows**: Only valid users, groups have appropriate membership, no blank pwds, Reasonable pwd policy, ‘strong’ passwords in use | OS-specific: local accounts vs domain accounts

* User account issues: Expiration dates, limit logon hours, special accounts: Admin/guest, Built-in accounts: IUSR/IWAM, TSInternetUser, HelpAssistant, SUPPORT

Audit activities: **PowerShell ActiveDirectory module** (Requires remote server admin tools RSAT, Win 10 and later), DSQuery/**DSGet**, PowerShell Get-Local\* cmdlets, Netwrix, ManageEngine, etc.

PS>Import-Module ActiveDirectory

Get-LocalUser | Select Name, Enabled | Where Enabled -eq $true

Get-AD\* commands

Get-ADForest, Get-ADDomain, Get-ADUser, Get-ADPrincipalGroupMembership

AD> **dsget**… (command)

**Orphaned User Accounts**: Unused accounts left on system, User left org, > 30 days, never logged in, delete unneeded accts

* With PowerShell and DSQuery

**Service Accounts** | Group Memberships | Windows Passwords | Passwords at Rest (SAM database or AD)

* LM hash (DES-based) | NTLM has (MD4) | Use **NoLMHash** to disable LM storage (Gorup policy setting)

Passwords in transit: Use **Kerberos** | NTLMv1 vuln to replays | NTLMv2 vuln to MITM

**Rights (privileges like “do”) and Permissions (access controls like read, add, modify, execute, delete)**: Protecting data at rest (restrict system access, limit rights/priv, enforce resource permissions, use encryption)

* RBAC should be used (Users inherit rights of groups)

**AGDLP**: Accounts -> Global Groups -> Domain groups -> Local groups -> Permissions (or rights)

**JEA** (Just Enough Administration): for specific individuals | Configuration in **.pssc file or .psrc** **file**

**PS> Get-AccountsWithUserRight -Right SeDebugPrivilege**

Windows NTFS permissions: “Deny” overrides “allow” | effective permissions are **sum total** of all permissions.

Windows Share Permissions

PS> **Get-SMBShare**

Get-ACL retireves info about access control lists on objects

PS>Get-Acl -Path C:\Windows\system32\ OR PS>Get-Acl HKCU:\Software\Microsoft\Windows

**File Integrity Assessment (FIA)**: Ensure key system or data files are not tampered with | Crypto hash (SHA256) to monitor

Tripwire (Paid), OSSEC (free, rely on SIEM) tools are FIA tools

**Windows Group Policy & Logging**: Manage Windows settings in a domain | Has no built-in audit capability, an **enforcement-only system**.

Audit Qs: Has security policy within the org been mapped to technical controls available within security templates?

**Audit Trail**: Ensure security logging is enabled and configured | Event viewer, Audit policy

*Event viewer*: Primary logging tool in Windows | Logons, Logoffs, Use of rights and permissions, access to objects, track processes | Configure log files: Size and Retention time/size and actions

*Audit Policy*: success, failed, or both are logged

**Query logon events with PowerShell**

PS>Get-EventLog security -InstanceId 4624 | select-object timegenerated, ….

Policy Change Event

**Log Management**: Objective: Ensure logs are cleared, rotated, and/or consolidated | Centralized | Continuous Aggregation | Must have SIEM (ELK, Splunk, ArcSight, SCOM/ACS)

**Auditing at Scale**: **CIS-CAT** | Wiin 10 and Ubuntu | **Fleet** (opensource by Kolide, Web/CLI for managing osquery across fleet of systems (tens of thousands), Fleet GUI has hosts, queries, schedules, policies, CLI,

**BOOK-3: Auditing Linux**

**System accreditation**: Good configuration means secure system and ease of auditing.

*Accreditation checklist*: Internal standards, CIS benchmarks, Cyber.mil STIGs and SRG (Security Req Guide)

System Description, Security Categorization, Threat identification, Vuln identification, etc.

System Information & File systems: Everything in Linux is a **FILE**.

Files, Directories, Input/Output devices, Disks (block devices), Running processes, Network sockets

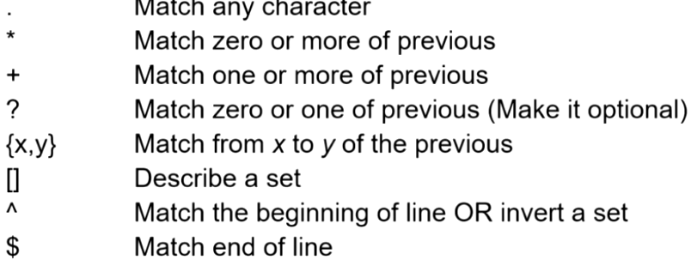
**Special directories**: Psuedo filesystems (ephemeral info): /proc [running processes], /dev [hardware devices], /sys or /proc/sys [kernel & system settings], /run or /var/run [Temp system files]

**$cat** (command) -> same as Microsoft **C:\>type** (command)

**$more, less, head, tail** (-f will follow file) | **man** (Unix manual) -k flag for keyword search | **apropos** (man for keyword search)

**$ls** (list) -> -a (all files) | -l (long and extended info) | -t (sort by timestamp) | -r (reverse sort order)

Slicing and dicing: $grep [limited regexp support, flags: -I, -v, -E, -a, -r, -c, -o, --help], $egrep [full regex support],

**$cut**, **$sed** [stream editor, remove unwanted text, convert, reformat],

(delete spaces start & end) 

Regex is greedy by default | Good for log analysis =====>

**$awk** [Aho, Weinberger, Kernighan] [pattern-matching]

e.g. $awk -F: `{print $1;}` /etc/passwd

**$cut, sort, uniq, wc**

**OS checklist**: Demographic info, Filesystem management, Linux system hardening, Services, Patching/package mgmt., Network config/hardening, User & priv mgmt., Logging and monitoring

$**lsb\_release** //Linux distribution | Kernel version: **$uname -a** | Kernel is the core of OS (ring 0)

Kernel settings file: $cat **/etc/sysctl.conf** and **/etc/sysctl.d/\***

Audit by viewing files in **/proc/sys** (running profile)

Ensure OS/user/application data are on separate partition: Handles availability issues;

Audit with **mount** and **mountpoints**

Check for use of modern filesystem drivers: **modprobe** and **lsmod**

Permissions: **User (owner) : Group : Other (world)**

read, write, execute, t (stickybit), s (SUID), S (SGID)

File markers: l (link), c (character), b (block for disk), p (pipe for inter-process), d (directory)

Permissions are octal | base-8 (0-7) | **Read=4, Write=2, Execute=1, SUID=4, SGID=2, Sticky=1**

**$chmod** modifies permissions; character notation for chmod: u (user), g (group), o (others), a (all)

**$find** (search directory for files with match)  (SUID or SGID set) **<- priv escalation files**

NFS/RPC: **$exportfs (**/etc/exports) | **Mounts: /etc/fstab, /etc/mtab** (*Remote filesystems can be checked here*)

**$rpcinfo -p** (to find NFS) | **$ps | grep -i nfs** | **$ps -xa | grep nfs**

Audit toolkit: **osquery for Linux**: osqueryi> select version, path from kernel\_info;

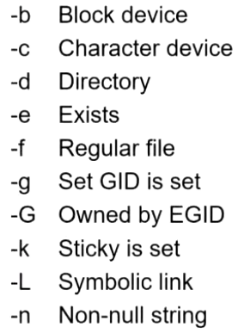
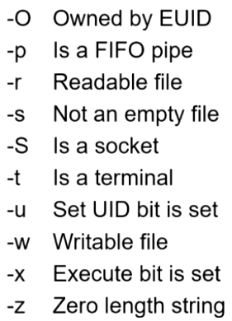
**osquery tables**: kernel\_modules, ssh\_configs, **suid\_bin** (executables with priv escalation)

**Bash scripting**: **$script** audit\_otuput.txt //script command takes notes for us // --timing flag to save a timing file

(type **exit** to come out of this script command)

Unix philosophy: Do one thing really well. Corollary: Don’t reinvent the wheel.

**Bash** (Bourne Again Shell) [Free] | **Korn Shell** (ksh) [HP/UX, AIS, Solaris, OpenBSD] | **Bourne shell** (sh) [Solaris, System 7]



Variables | echo | If/Then and Brackets | Square Brackets and test |

Comparisons: -gt, -le, -ge, -lt, -ne, -eq, !=

A -nt B (A newer than B) | A -ot B (A older than B) | A -ef B (A and B are linked)

Arguments | Accept user input | Functions

**Advantages of Scripting:** Conducting audit in same way every time,

Results and reporting automated, Simplifies analysis.

**System hardening**: Use mandatory access control | Enable eXecute Disable/No eXecute (XD/NX) protection

Turn ON ASLR | Use File integrity monitoring software.

Mandatory Access Control: **AppArmor** (Augments Discretionary access control by file paths) | Ensure AppArmor is not disabled in boot loader (**apparmor = 0** means disabled) | $apparmor status or **$sudo aa-status**

**SELinux**: Uses traditional MAC techniques | Objects and processes given labels (security contexts) | Ensure not disabled (selinux=0 means disabled in boot loader) | Use **$sudo sestatus** | check **/etc/selinux/config** for ‘SELINUXTYPE=targeted’

**Buffer-overflow protection**: XD, NX | grep for ‘Execute Disable’ on journal/dmesg | ASLR

**Disabling Hotkeys**: Control+Alt+Delete should not automatically reboot |

$ln -sf /dev/null /etc/system/system/ctrl-alt-del.target

File Integrity Assessment: **Tripwire** (Paid and Free[no GUI]) | **AIDE** (Advanced Intrusion Detection Env, CIS benchmarks) | **OSSEC** (free)

$tripwire --init //if you get error, that indicates file systems don’t exist.

Then edit tripwire policy to comment out files that don’t exit

Convert text based policy into binary: $twadmin -m P /etc/tripwire/twpol.txt // creates **/etc/tripwire/tw.pol**

**Tripwire and Change Control**

Need a ‘clean’ report of tripwire indicating zero violations.

**RPM** for file integrity: Only for packages it installed | $rpm -V <package\_name> | $rpm -V sendmail | $rpm -Va //for all

**$rpm -Va** will report **rootkit** is still correctly installed

**Services, Network config, and logs**: Linux **Startup services** are launched by the **‘init’** system. **$ls -l /etc/rc2.d/**

Inetd/xinetd (**Internet super server**)

**$systemctl** to view services | $systemctl list-units (configured services) | $systemctl -a (all services)

**$netstat** (active connections and listening ports) | with or without process information | **$lsof -i -n | grep LISTEN** (process)

**$nmap** -sT -sV T:1-65535 10.50.7.20

**Vendor patching**: AIX: oslevel and oslevel -s | Linux: Satellite server/Spacewalk | Ubuntu: Aptitute | Oracle: **Patch Check Advanced (PCA)**

Ubuntu: *$apt update* | Red-Hat: *$yum check-update*

*Supplemental patch data gathering*: **patch velocity (frequency), patch age (how long ago)**

Non-package daemons: check for those manually installed

**Kernel patching**: Reboots not always required: **LivePatch, Kpatch, Ksplice**

Configuration management: **Ansible**(imperative), **Puppet** (declarative/agent-based), **Chef** (declarative), **SaltStack** (both)

**Kernel network settings**: not well documented | CIS benchmarks and DISA STIGs are good sources | protects from MTIM, DoS

Check for: IP forwarding, Source routing, ICMP redirects/IPv6 router advertisements, Ignore broadcast ICMP, Syn cookies

**TCP wrappers**: library: **libwrap.so**

**Access Control**: ACLs /etc/hosts.deny | /etc/hosts.allow (by default deny ALL)

**Iptables**: Host-based IP firewall | Policies for chain (I/p, O/p, Forward) should be configured to DROP traffic (default deny)

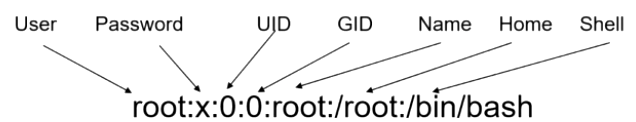
**Auditd**: Kernel based auditing | enabled with **$auditctl** | text-based logs: **$ausearch**, **$aureport**

**System Journal**: $journalctl: query the journal files for entries | flags: -f (real time), -b (system boots), -u (current user), -S (since date/time), -U (until date/time) | **$systemd**

**Unix logs**: Log file: /var/run/utmp (current logged in user, ephemeral, **$who -u**) | /var/log/wtmp (binary file, username, historical records, **$last**) | /var/log/btmp (**$lastb -adx**) | /var/log/messages (or syslog, **/etc/syslog.conf**) | /var/log/secure

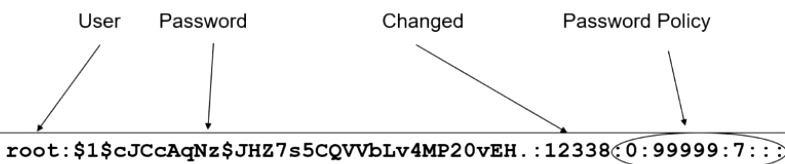
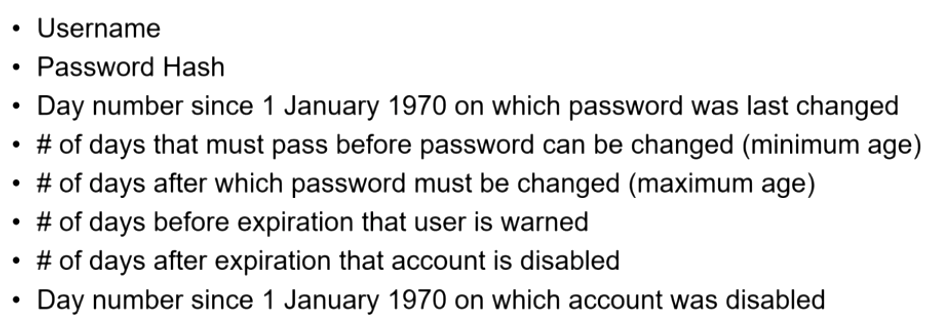
Contains: snapshot, login-logout history, bad login history, messages from syslog, access & authn info

**Log rotation**: /etc/logrotate.conf | /etc/logrotate.d/\* (like /etc/logrotate.d/mail.log) | Unix logs auto-rotate



**User and privilege management**: /etc/passwd file

/etc/shadow (only root)

**Password policy**: /etc/login.defs

Password crackers: **John The Ripper** (Distributed cracking, Runs of Win & Unix, BSD-style pwds, DES-based pwds, Twofish-based pwds, NTLM hashes)

**Pluggable Authentication Modules (PAM)**: Kerberos, Active Directory, password histories, lengths, smart card

**/etc/pam.conf** and **/etc/pam.d/**

PAM modules: **Pam\_cracklib**: check password strength | **Pam\_pwquality**: /etc/security/pwquality.conf | **Pam\_tally2 and Pam\_faillock** (for account lockouts)

**Limit Root Access**: **/etc/securetty** (console or telnet) | Local root account disable: **$passwd -l root** (pwd locked, ssh works)

/etc/sshd\_config and /etc/ssh/sshd\_config.d/ (600 perms, private keys: 600, public keys: 644)

ssh test: **$ssh -v localhost**

Priv escalation: **$sudo find / -perm /6000 -type f** //file files with SUID & SGID programs owned by root

**$su vs $sudo**

**Jump servers**: maintains privileges creds for systems | Log all activity , even a video |

**Full system audits**: **Lynis**: Host-based assessment | AIX, FreeBSD, HP-UX, Linux, OS X, NetBSD, OpenBSD, Solaris

Lynis audits based on discovered services and configuration

**Chef Inspec**: Compliace as code tool | JSON output | profiles based tests | Ruby | Dev-Sec.io, MITRE | Windows, Linux, VMWare, Cloud providers

**CIS-CAT**: Lite version support Win 10 & Ubuntu

Vulnerability scanners: Credentialed scans | **Nessus**

**BOOK-4: Auditing Cloud Infrastructure**

**Private Cloud Infrastructure**: Hypervisors [OS virtualization] vs Containers [Application virtualization]

* Type-1 Hypervisor: bare metal (very thin virtualization layer to manage physical resources)
* Type-2 Hypervisor: hosted (software app running within a hosting OS)

Storage Area Network (SAN) | **Common issues found in Audit**

**VMWare-Specific**: vCenter Server (manages ESXi hypervisors [these host VMs]), RBAC, vSphere Hypervisor

**Web client/API/power CLI -> vCenter Server (VCSA) -> ESXi Server -> VM**

VMware software version is by “**build number**” (website has KB of versions)

**CIS VMWare Benchmark**: Install/patching, Comm, Logging, Access, Console, Storage, vNetwork, VM configuration

VMWare patches are installed VIBs (**v**Sphere **I**nstallation **B**undles) | List patches on host by PS>**Get-ESXCli** cmdlet.

VMWare host user: **PS>Get-VMHostAccount** | VMWare Host configuration: **PS>Get-VMHost**

**ExtensionData.Config** contains may config items.

VMWare Service Status: PS>**Get-VMHostService** | NTP & Syslog: PS>**Get-VMHostNtpServer** & **Get-VMHostSysLogServer**

VMWare **Lockdown Mode**:

* Normal (DCUI, direct console user interface, only for admin users on exception list; default no console for everyone)
* Strict (no admin, except thru vCenter; need to reinstall of vCenter access is lost; DCUI is disabled)

**VMWare Audit Tools**: Web UI (auditors are granted read-only), SSH (not appropriate for auditors), VMWare API, **PowerCLI**, As Built Report PowerShell modules (3rd party; preinstalled on Win 10 VM), Robware **RVTools** (Win GUI to inventory info on VMware infra; export as CSV or Excel)

* PowerCLI is recommended for querying entire data center | Save setings to file for later analysis

**ESX CLI** -> Every CLI command is exposed as a function.

**Containers**: Microservices | Orchestration (K8s, EKS, Docker swarm, Azure Service Fabric, Helios by Spotify)

Pods:- Group contains to run on same physical host

Services:- Public container services to know IP:port sockets | located using K8s service discovery

Volumes:- provide storage to containers (persistent or local ephemeral [/var/log])

Namespaces:- provide isolation for related pods, replication controllers, and volumes

Avoid using “default“ namespace for deployments.

**Auditing Kubernetes**: Security of config files, API server, controller manager, Scheduler, Etcd security, Authn/Authz, Logging

(CIS benchmark for K8s)

Worker node configuration: config file security | kubelet settings

Security policies: RBAC for service account, Pod security, Network and CNI (container network interface), Secrets mgt

**Docker**: CIS docker benchmark: Host OS system settings, Docker daemon config, config file security, Image file security, config of runtime env, Operations: image/container sprawl, Docker swarm

**Serverless Containers**: Even containers can be overkill for a single function | AWS Lambda, Azure Functions, Google Cloud Functions, Knative

Best practices: Inventory (tags, watch code repo), Limit to single IAM role, security in deploy pipeline, Access control, logs

**Cloud Audit Toolkit**: Cloud Custodian | Prowler | ElectricEye | CloudQuery.io | Chef Inspec

* **Cloud Custodian (c7n)**: opensource, CNCF, monitoring and enforcement in AWS, Azure, GCP
  + Policies written as YAML file (need to validate policy file before running c7n, **resources.json** is created)
  + Can perform actions on non-compliant resources (tag, policies, send notif, stop instances, change settings)
* **Prowler**: AWS, Azure, GCP security monitoring | Python based | Not fully mature for Azure & GCP | more for AWS
  + 350+ checks includes CIS benchmark, HIPPA, PCI, ISO, SOC2, GDPR, etc.
  + Local binary or container
  + Outputs to **security hub** (JSON-ASFF, Amazon security finding format), S3 bucket (JSON, CSV, HTML, color or mono text), Junit XML to Amazon codebuild, Local files
* **ElectricEye**: Compliance testing | Python | AWS, GCP, Oracle Cloud, ServiceNow, Microsoft 365 Enterprise
  + 900+ checks (scripts containing checks)
  + Integrates with PostgreSQL, Security Hub for alerting and automated remediation
  + Can supplement with **Shodan** queries
* **CloudQuery.io**: SQL based | fetches data from multiple cloud and saves in local or cloud DB | pre-built policies for AWS, Azure, GCP, CIS benchmarks | Comes with **Grafana dashboards** for resource inventory and compliance stats
  + 50+ CSPs (Alibaba cloud, Cloudflare, CrowdStrike, Datadog, etc.) <- **useful for CSPM**
  + Policies are in GitHub repo
  + Win, Linux, macOS binaries on GitHub | **cloudquery sync** fetches resources defined in YAML

**Auditing Public Cloud**:

Shared responsibility model | CIS benchmark mappings

IAM: Limit admin access, check 2FA, Restricted API access keys, Login monitoring

IAM Objects: Users, Groups, Roles, Policies

Root account: Shared rarely be used on CSP login; Enable MFA

**AWS Password Policy** | **Azure IAM** (Entra ID): **Microsoft.Graph PowerShell modules** can query many of the settings |

**GCP IAM**: Google workspace admin console, 8 to100 chars., Google only recommends not using personal Gmail accounts to corporate accounts.

**PowerShell for AWS IAM**: PS>Get-IAMUserList, Get-IAMRoleList, Get-IAMUser, Get-IAMAccountPasswordPolicy, Get-IAMMFADevice, Get-IAMAccountSummary, Get-IAMAccessKey

**Azure** has multiple admin types | create admin users with least privilege | **GCP** has org admin role, ensure services accts are not in that role

Audit AWS: **$aws iam get-account-summary** to confirm that *root account has no access keys* and to verify MFA.

For access keys: **$aws iam generate-credential-report**

Audit Azure: **$az role definition list** and **$az role assignment list** users must be in lowest priv roles

For MFA: check MULTI-FACTOR AUTH STATUS for all users in console.

For access keys: no direct check in benchmark

Audit GCP: manually validate that no service accounts have any *admin* roles.

For MFA: **$gcloud organizations get-iam-policy**

Enable MFA for admins.

AWS Access keys: LIMIT ONE ACCESS KEY PER USER. (90 day rotation)

**Logging & Monitoring**: Writing logs to storage account/bucket; Logs are needed for forensics; Monitoring involves using logs to produce metrics and alerts for key events;

AWS logging: **CloudTrail**: Can log calls to AWS API, all console actions, CLI | PowerShell commands use API

For **risky** events in API calls, use **log metric filters**

Logs are saved to S3 buckets | Can be sent to **CloudWatch** for further analysis

* CloudTrail is intended for governance and risk auditing | Useful for IR and Forensics

**CloudWatch**: Application/System logging; health and performance; Built for managers, SREs

**SNS**: Simple Notification Service (A2A, app to app | A2P, app to person)

For CIS benchmark validation: use $aws cloudtrail, config service, S3 logging, VPC flow logging

$aws cloudtrail describe-trails -> check existing trails

$aws cloudtrail get-trial-status -> ensure logging is enabled

$aws cloudtrail get-event-selectors -> see the event types being logged

$aws sns list-subscriptions-by-topic -> verify alarms are sent to SNS service

Azure logging: Uses log profiles for control and mgmt. on Azure subscription | Configure it to log all regions, and global.

Default storage is 90 days; increase to 1 year or more. Logs sent to storage containers: remove public access and encrypt it.

$az monitor log-profiles list -> check that a profile exists

$az storage container list -> validate settings for storage container (ensure ‘publicAccess’ is NULL)

$az storage account list -> check encryption settings for container

keySource to be ‘Microsoft.Keyvault’ and keyVaultProperties to be ‘non-null’

Create **alerts** for high-risk activity subscription: policy assignments, network security group changes, NSG rule changes, security solution changes, SQL server firewall changes, update to security policies

*Validate monitoring in Azure*: by console or CLI | Rule location set to ‘global’, scope for entire subscription, enabled = true

GCP logging: Cloud Audit Logs service, sends events into storage buckets | **Log sinks** define the buckets that will be used | Export copies from buckets are sent to SIEM | **Bucket lock** to protect against overwriting/deleting.

Monitoring: Changes to project ownership, Audit configuration, custom role, VPC config, cloud storage permissions, SQL instance config.

*Validate monitoring in GCP*: **$gcloud organizations get-iam-policy**, **$gcloud resource-manager folders get-iam-policy**, **$gcloud projects get-iam-policy**.

* Check log sink policies: **$gcloud logging sinks list**

**Networking and Infrastructure**:

Look for default-deny settings on subnets and VPCs | ensure least privilege access between subnets and VPCs | disallow ingress to admin ports like SSH and RDP | enable logging and monitoring for network traffic

AWS VPC: Container for EC2 instances | Assign IPv4 CIDR block at creation | Spans all availability zones in region | subnets

**Best practices**: Default deny rules on all NAC lists | No unrestricted (0.0.0.0) to admin ports like SSH/RDP | No unrestricted access to database servers, including NoSQL databases | Allow only required ports to/from internet and internally.

**Validate network settings:**

$aws ec2 describe-network-acls // check appropriate rules

$az network nsg list // review rules

$gcp compute firewall-rules list // check rules

**Compute**: Not heavily covered by benchmarks

**AWS IMDS**: protects against vulnerabilities | available at REST API using private IP: 169.254.169.254

IMDSv1 – no security tokens to access

IMDS stores credentials for accessing other resources (e.g. S3 buckets) | Audits must push for use of **IMDSv2**.

AWS central management for compute resources: AWS System manager (Fleet manager, Session manager, Patch) | Config

* AWS operations hub core features:
  + Ops mgmt., App mgmt., Change mgmt., Node mgmt.

**AWS Fleet manager**: mgmt. of cloud-based and on-prem resources | Monitor health and perf issues (CPU, Memory, Disk, Network) | Mgmt of Windows, Mac, Linux for Filesystems, Registry/Settings, Users & perms

**AWS Session manager**: Evolution of cloud host security

Worst: allow direct access to admin ports (SSH, RDP)

Better: build hardened bastion host

Best: Connect using session manager – browser based (Shell, CLI, RDP)

Auditing to S3 or CloudTrail

**AWS Patch manager**: OS and App patches from central console (EC2 or on-prem); Uses “**patch baselines**” to auto-approve based on severity, OS or app, allow/disallow list.

**AWS Config**: Continually assesses, audits, evaluates the configurations and relationships of your resources | *Resource inventory is hard in the cloud* | uses rules for assessing configuration | monitors on its own or with security hub.

**Microsoft Defender for cloud**: Azure CIS benchmark as a full section on this | cross-platform | Azure VMs, AWS instances, On-prem systems, Databases, Containers | Single dashboard for all

**Infrastructure as Code**: Deploy entire cloud in DevOps env | AWS CloudFormation, **Terraform** | scan for common flaws before deploying; | Terraform uses *HashiCorp Configuration language* (HCL) | Build modules, use variables, create output files

AWS CloudFormation uses templates written in JSON or YAML | builds “**stacks**” of resources

**IaC Static Analysis**: CloudFormation Guard (from AWS), **cfn\_nag** (from Stelligent), **Terrascan** (from Tenable)

**Storage and Databases**: No public access (least-priv for users, Default-deny rules on network), Use encryption in transit and rest, Encrypt storage for VMs and DBs

**AWS S3**: buckets have unique URLs | per region | REST API access to data, via console, or 3rd party tools

Misconfigured bucket permissions is common security problem | “block public access” must be set to TRUE.

Validating storage in AWS: Check EC2 default encryption settings. | **$aws rds describe-db-instances** to check encryption settings for DB storage in RDS | **$aws s3api get-public-access-block –bucket <bucket\_name>** //to check block public Acls.

$aws s3 ls and $aws s3api get-bucket-encryption -> validate encryption

$aws s3api get-bucket-policy -> verify HTTP access is disabled

$aws s3api get-bucket-versioning -> check MFA is required for deletes or risky changes on buckets

$aws s3api get-public-access-block -> ensure public access is disabled on buckets

Validating storage in Azure: **$az storage account list** -> HTTPS connection, public access deny check, default-deny check

$az monitor activity-log //access key rotation

$az storage logging show //validate logging for storage requests

**Databases**: Well covered in AWS and GCP benchmarks | most controls are same as in on-prem | log, encrypt, access

**Beyond the benchmarks**: IMDS and EC2 instances security is not covered in benchmarks

**AWS compliance tools**:

* **AWS Security hub**: built-in checks for PCI, CIS. | consolidates mgmt. and monitoring info from AWS + 3rd party | ASFF format for security findings | across all regions, can consolidate into one region
* **Amazon Inspector**: Vuln mgmt. for workloads | Uses agent installed on EC2 instances | scans for s/w vul; Send to Security Hub for analysis
* **AWS IAM Access Analyzer**: Identify resources accessed from external entities | single region | check access for S3, IAM roles, KMS keys, Lambda functions, Secrets Manager secrets

Look for **well-architected frameworks, Landing zones** (pre-provisioned env approved by orgs; forces new infra to be created inline with standards; define guardrails to ensure approved config/apps/policies are used)**, Google “shared fate”** [put provider and customer on same team] to get guidance on outside-CSP responsibility works.

**Evaluate entire cloud usage**: Cloud Adoption Frameworks (**CAFs**) | Google adoption framework/maturity model | **AWS** security maturity model (*Four maturity*: Quick wins, Foundational, Efficient, Optimized; 9 CAF categories) | **VMWare** cloud maturity model (Visibility -> Optimization -> Governance & Automation -> Business integration) | SANS **LDS520** maturity framework (Initial -> Managed -> Defined -> Quantitatively managed -> Optimizing;8 dimensions) | **AUD507** cloud compliance roadmap (3 phases: Inventory/Visibility -> Foundational -> Automation/Integration)

**CAFs** tend to focus on migration to cloud (more marketing)

AWS: very product-focused | Azure: migrate workloads to Azure | GCP: migration-focused but widely usable

**Google Adoption Framework/Maturity Model**: SRE, BeyondCorp **zero trust model**

Learn -> Lead -> Scale -> Secure | Tactical -> Strategic -> Transformational

**Azure Maturity Model**: Foundational -> Intermediate -> Advanced

**Cloud Security Posture Management (CSPM)**: Single pane of glass | can be expensive | understand blind-spots

**BOOK-5: Auditing Web Applications**

HTML: Describes how to paint a page | HTTP/2 (RFC 7540, May 2015), Based on Google SPDY, Binary

Request verbs: GET, POST, HEAD, PUT, DELETE, CONNECT, OPTIONS, TRACE, PATCH

Response codes: 100: Information/Upgrades | 200: success | 300: redirects | 400: client error | 500: server error

**WebDAV**: distributed authoring and versioning; uses HTTP to manage files on server; has verbs: COPY, MOVE, LOCK, UNLOCK, MKCOL, PROPFIND, PROPPATCH

SOA | SOAP | **RESTful APIs** (PUT, PATCH, DELETE) | AJAX (big problem: Bypassing authn or authz system with “bolt-on” AJAX) | **SPA** (AngularJS, Ember.js, React.js, Meteor.js): Only has <script> tags but **not <form> tag** | **Cookies** (HTTPOnly, Secure flag, Persistence, SameSite for CSRF) | CSS | OWASP | BurpSuite

OWASP Top 10 | OWASP Top 10 proactive controls |

* Server configuration, Secure dev practices, Authn & Access control, Data handling, Logging & monitoring

Directory Indexing | Server headers | Netcraft site report | Robots.txt | HTTPS / TLS | OCSP | **CAA (Certificate Authority Authorization) DNS records** | HSTS / SSL striping | Encryption | TLS ciphers (SSLyze Python tool, nmap, qualys) | Same Origin | CORS | CSP

OWASP ASVS | misuse cases | Complete Mediation (ESAPI, Single API for everything) | Libraries up-to-date | Wappalyzer | **Retire.JS** (identify outdated JS libraries in a site; browser plugin, CLI, Burp/ZAP plugin) | Error handling | Errors in SPA (JavaScript console) | Unexpected error handling | Code review | SAST | DAST (fingerprint first) | Don’t trust server headers | Burp Scanner | Automation | Business logic flaws

HTTP basic auth | Form-based | client-side certificates | **WebAuthn (passkey) uses public/private keypair** (serverside theft, phishing, brute-force protection), **Passkey registration** flow | **Passkey authn** flow | RESTful APIs **(JWT, API key)** | Username harvesting | Brute Force DoS / Account lockouts | Password hashing | **NIST recommendations SP800-63B**: do not use pwd from known breaches and *haveibeenpwned* database | Session tracking | URL rewriting | **Cookies** | Basic Authn (login and logout) | **Attack methods on session tracking** (predict/guess, eavesdrop, steal) | Controls for session tracking: Robust session ID, secure session tracking mechanism in traffic), finding patterns in session | **CSRF** (with POST, how to protect) |

**Data Handling**: SQL Injection | Dynamic SQL Injection | Don’t do input filtering for SQLi and DON’T do stored procedures |

To fix SQLi: **Parameterized** (or bound) **queries** | SQLMAP tool | Injection flaws: SQLi, LDAPi, XPATHi, HTMLi, | **XSS** (Persistent (or stored), Non-persistent (or reflected), DOM-based can be either) | **SSRF** |

**Anti-Caching** / prevent caching data on client side (use HTTP Expires header, or in HTML with “no-cache”)

**Logging and Monitoring** | Injection Flaws