1)

a)

i) Use nmap –sS –T5 –p- 10.39.26.128-159

ii) Use netcat [host] [port]

Port knocking is the process of contacting a pre-determined list of ports, with the final contact returning the port to connect to for the real service.

b)

i) Common PHP vulnerabilities:

* Taking user input and executing it as a shell command (use stuff like escapeshellarg())
* Taking user input and executing it as a DB query (Use prepared statements)
* Taking user input and including it in HTML (Use whitelist)

ii) I assume some exploit lets you execute shell commands, then cat-ing the file.

Use UDP for faster attempts?

2)

a)

i)

Passive:

* Study the source code (without visiting the site)
* Find DNS/Domain info
* Study open source code used by target

Active:

* Use dig to query the DNS server
* Traceroute
* Identify subnet addresses
* Scan open ports
* Identify target OS
* Study information sent by server
* Send random info and study services

ii) User could use a weak/easily guessable password (or have a hint that is easily guessable) that leads to the attacker performing online dictionary attacks or otherwise guessing the password to spoof.

Hosting domain may not use HTTPS - requests could be intercepted, read or tampered with (information disclosure).

Browser extensions can contain scripts that allow them to forward information to the creator. Similar for third party domains that have content on the page.

ii) User: May have login information leaked/stolen, by a third-party domain script or browser extension. Comes under I of STRIDE (Information disclosure)

Hosting domain: Could have vulnerability that allows SQL injection by user to delete a database. Comes under T of STRIDE (Tampering)

Third-party domain: ? Man-in-the-middle, DDoS (Maybe Denial of Service by not storing the domain in the DNS server)

Browser extension: Could be used as way to track user by hosting/third party domain? Which part of STRIDE would this be?

Browser extension: Since content script can read and modify DOM of pages, it can intentially add malicious contents / advertisement on pages without user consent (Tampering)

b)

ii) Unknown, could be a JS exploit

iii) JS or PHP exploit? Send every visitors cookie to you? Communication is over HTTP?

3)

a)

i) Path traversal mitigation: Web server has a special account that can only access public files, or sandboxed to a virtual file system (chroot jail)

Remote file inclusion mitigation: Whitelist of allowed inputs to limit input files to known choices

Server-side request forgery: Prevent user being able to include URLs (secure regex?) Or whitelist server requests, or don’t handle unexpected responses

ii) Security measures for:

* Invalid username: Don’t report that username is invalid, just say combination of user/pass not valid
* Few failed attempts for username: Request captcha, ban IP for limited time, email user with warning, delegate to another device (2-factor authentication). Limit frequency of requests for IP
* Successful login: Send randomised access token/cookie? Force user to connect using HTTPS? Provide details of all previous login attempts (e.g., date and time of last unsuccessful login attempt); check if fingerprint is the same as previous times, otherwise email user

b)

i) Some form of URL/../link/to/vulnerable/resource ?

Check hr.borkbork.co.uk/private/logins.txt

ii) SQL injection common steps:

* Attempt simple SQL injection: OR 1=1
* Use # to remove any code after what you enter (MySQL)
* Use: “‘ order by 1 #” and increment value to determine how many columns are being displayed
* Union select user, password from TABLE\_NAME
* Select \* from information\_schema.tables
  + Alternatively: ’ union select table\_name from information\_schema.tables where table\_name like “user%” #
* Find out column names:
  + ‘ union select column\_name from information\_schema.columns where table\_name = <the table that you found in the previous step> #
* Perform boolean queries (maybe this takes too long to be used in the exam)

4)

a)

i) Full disclosure (Grey hat + security researchers):

* Bring public scrutiny on the company, can force them to fix the issue
* However, could allow attackers to exploit the vulnerability before fix.

Responsible disclosure (White hat - pentester, or software vendors):

* Affected party has time to patch/fix exploit, before releasing statement
* Can take a long time depending on the laziness of the party

No disclosure (Black hat):

* Attackers who don’t know vulnerability aren’t able to use it
* Vendor may not know, and attackers may have figured it out and exploited already

ii) Only target certain users: Check fingerprint, send malware if matches (e.g. location, IP, user agent), otherwise behave like normal.

Limitation: Users may use VPN/spoof fingerprint, depending on stability/precision trade-off, you may end up targeting extra victims or the fingerprint could be invalid by the time malware checks for fingerprint match

Mitigation: Spoofing your fingerprint? Anti-virus firewall, IDS/IPS, packet filtering

Spear phishing: only send the link to targeted users matching the profile you want, make sure the link is long and complicated enough that web users don’t accidentally stumble onto it e.g. from mistyping a domain name

* Limitations: need in-depth research to find the emails of users that fit the profile you want, more effort than just blasting the link to random databases of users
* Mitigation: don’t click on suspicious links

b)

i) Use burp to intercept request to see what it actually sends?

ii) Spoof own browser? Set user agent = “President”?

iii) ?