DUMPSBOSS.

CompTIA PenTest+ Exam

CompTIA PT0-003

Version Demo

Total Demo Questions: 15

Total Premium Questions: 252

Buy Premium PDF

https://dumpsboss.co

support@dumpsboss.co

support@dumpsboss.co
dumpsboss.co

QUESTION NO: 1 - (HOTSPOT)

HOTSPOT

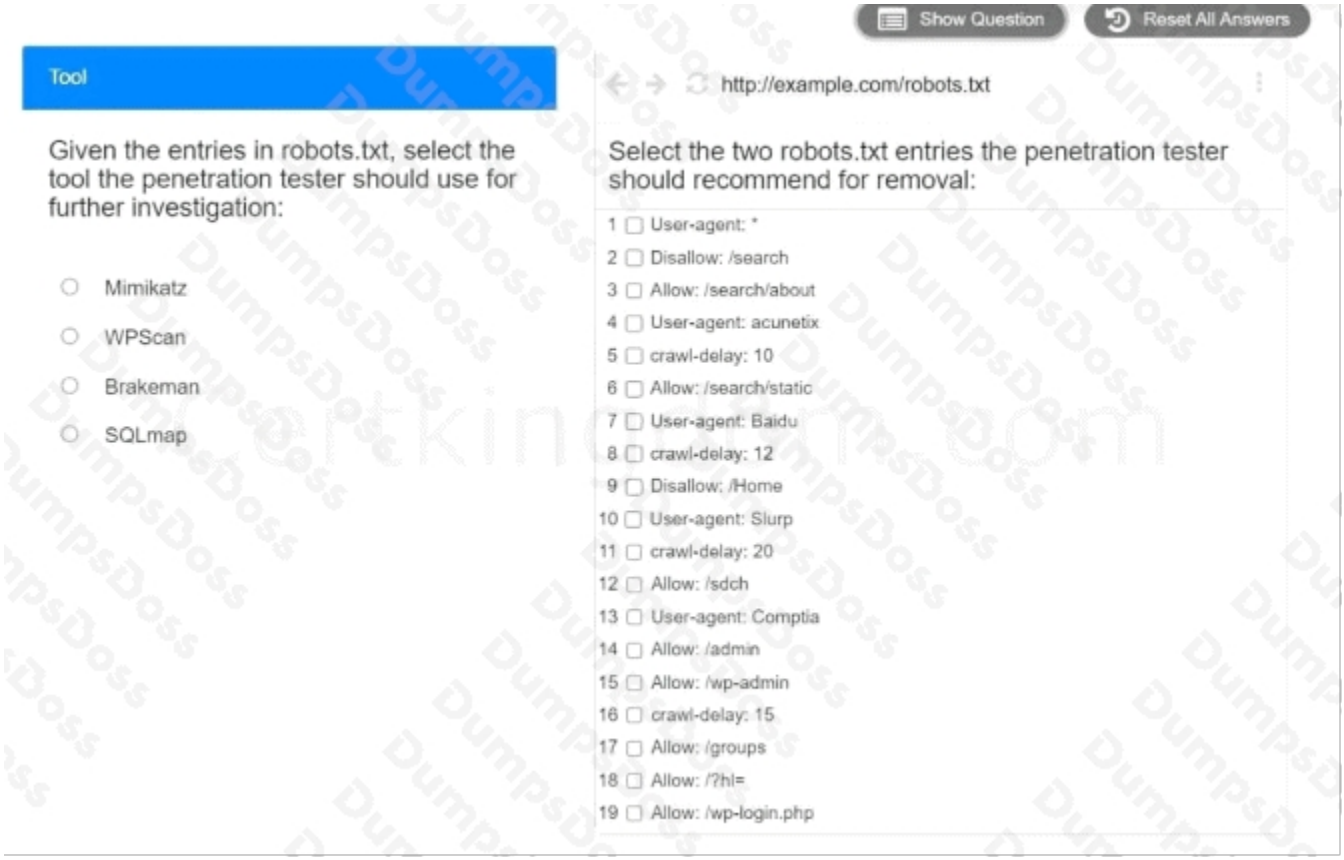
[Information Gathering and Vulnerability Scanning]

A penetration tester is performing reconnaissance for a web application assessment. Upon investigation, the tester reviews the robots.txt file for items of interest.

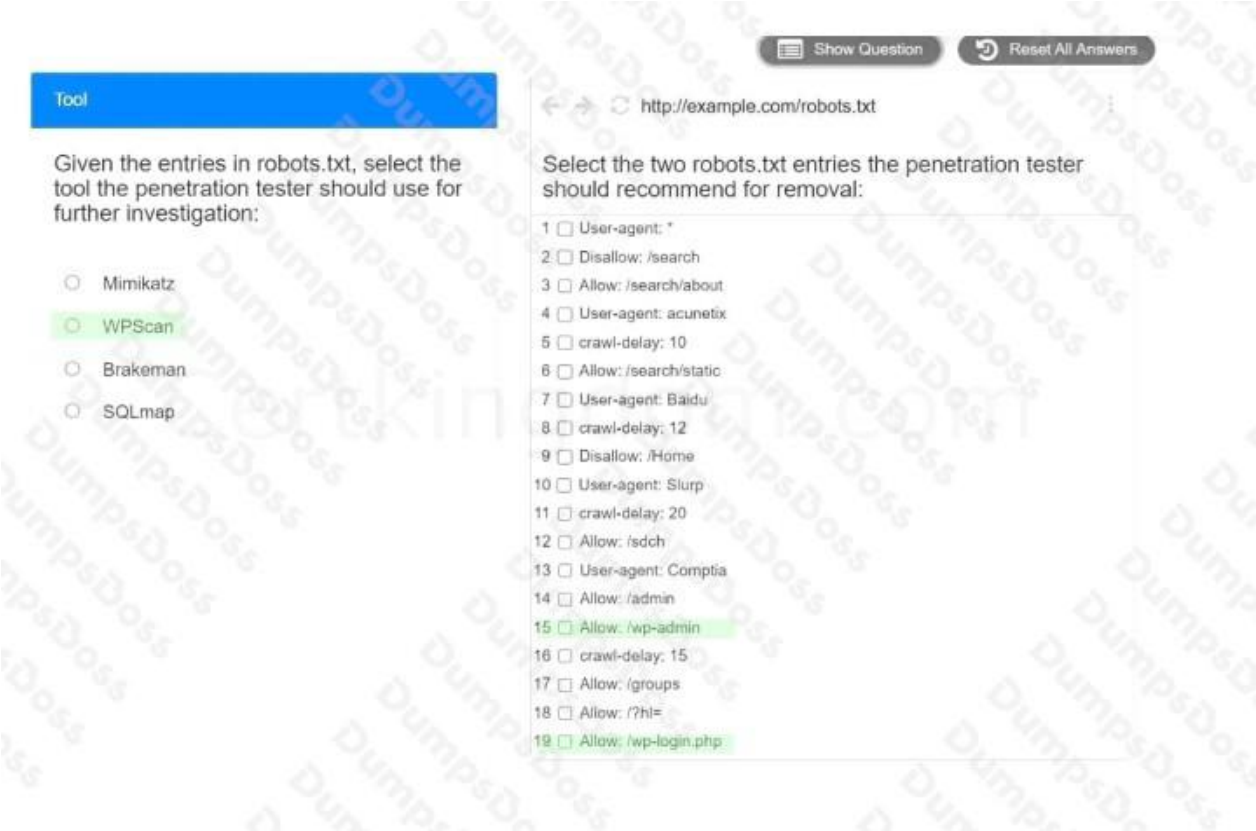
INSTRUCTIONS

Select the tool the penetration tester should use for further investigation.

Select the two entries in the robots.txt file that the penetration tester should recommend for removal.



ANSWER:



Explanation:

Looking at the robots.txt content, the biggest clue is that it contains WordPress-specific paths. In particular, the file explicitly lists **Allow: /wp-admin** and **Allow: /wp-login.php**. Those are strong indicators the target is running WordPress (or at least exposing WordPress endpoints). Because of that, the most appropriate tool to use for deeper investigation is **WPScan**, since it's purpose-built for WordPress reconnaissance and vulnerability scanning (enumerating users, plugins, themes, and checking versions against known CVEs). Tools like Mimikatz are for credential extraction on Windows hosts, Brakeman is for Ruby on Rails code scanning, and sqlmap is focused on SQL injection testing rather than WordPress enumeration.

For the two robots.txt entries to recommend removing, you want to focus on lines that unnecessarily advertise sensitive administrative or authentication functionality. While robots.txt is not an access control mechanism, it's commonly used by attackers as a "map" of interesting locations. In this list, the two best candidates are **Allow: /wp-admin** and **Allow: /wp-login.php**. Publishing these paths makes it easier for an attacker to quickly find the admin dashboard and the login endpoint for password spraying/brute force attempts. Removing them doesn't secure the endpoints by itself, but it reduces passive information disclosure and lowers the chance of opportunistic automated scanning finding them immediately.

For more background, see the WPScan project page (https://wpscan.com/) and OWASP guidance that robots.txt should not be relied on to protect sensitive resources (https://owasp.org/www-community/attacks/Robots_Exclusion_Standard).

QUESTION NO: 2 - (SIMULATION)

SIMULATION

[Attacks and Exploits]

A previous penetration test report identified a host with vulnerabilities that was successfully exploited. Management has requested that an internal member of the security team reassess the host to determine if the vulnerability still exists.

Which of the following commands would **most** likely exploit the services?

- medusa -h 192.168.10.2 -u admin -P 500-worst-passwords.txt -M rpcbind
- hydra -l lowpriv -P 500-worst-passwords.txt -t 4 ssh://192.168.10.2:22
- crowbar -b rdp -s 192.168.10.2/32 -u administrator -C 500-worst-passwords.txt -n 1
- ncrack -TS -user lowpriv -P 500-worst-passwords.txt -p telnet -g CL=1 192.168.10.2

Part 1:

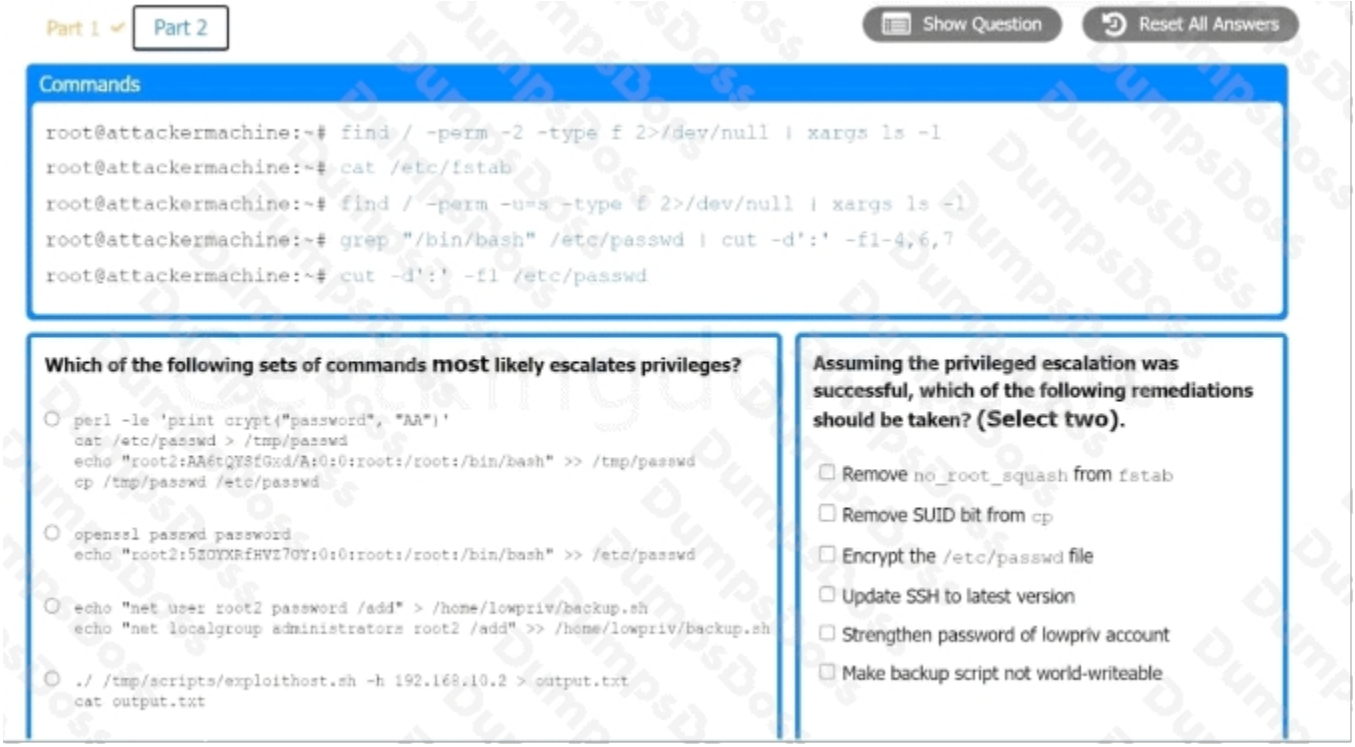
. Analyze the output and select the command to exploit the vulnerable service.

Part 2:

. Analyze the output from each command.

Â· Select the appropriate set of commands to escalate privileges.

Â· Identify which remediation steps should be taken.



ANSWER: Seethebelowforcompletesolution.

Explanation:

The command that would most likely exploit the services is:

hydra -l lowpriv -P 500-worst-passwords.txt -t 4 ssh:/.168.10.2:22 The appropriate set of commands to escalate privileges is:

echo "root2:5ZOYXRFHVZ7OY::0:0:root:/root:/bin/bash" >> /etc/passwd The remediations that should be taken after the successful privilege escalation are: Remove the SUID bit from cp.

Make backup script not world-writable.

Comprehensive Step-by-Step Explanation of the Simulation

Part 1: Exploiting Vulnerable Service

Nmap Scan Analysis

Command: nmap -sC -T4 192.168.10.2

Purpose: This command runs a default script scan with timing template 4 (aggressive). Output:

bash

Copy code

Port State Service

22/tcp open ssh

23/tcp closed telnet

80/tcp open http

111/tcp closed rpcbind

445/tcp open samba

3389/tcp closed rdp

Ports open are SSH (22), HTTP (80), and Samba (445).

Enumerating Samba Shares

Command: enum4linux -S 192.168.10.2 Purpose: To enumerate Samba shares and users. Output: makefile Copy code user:[games] rid:[0x3f2] user:[nobody] rid:[0x1f5] user:[bind] rid:[0x4ba] user:[proxy] rid:[0x42] user:[syslog] rid:[0x4ba] user:[www-data] rid:[0x42a] user:[root] rid:[0x3e8] user:[news] rid:[0x3fa] user:[lowpriv] rid:[0x3fa] We identify a user lowpriv.

Selecting Exploit Command

Hydra Command: hydra -l lowpriv -P 500-worst-passwords.txt -t 4 ssh:/.168.10.2:22

Purpose: To perform a brute force attack on SSH using the lowpriv user and a list of the 500 worst passwords.

-l lowpriv: Specifies the username.

-P 500-worst-passwords.txt: Specifies the password list.

-t 4: Uses 4 tasks/threads for the attack. ssh:/.168.10.2:22: Specifies the SSH service and port.

Executing the Hydra Command

Result: Successful login as lowpriv user if a match is found.

Part 2: Privilege Escalation and Remediation

Finding SUID Binaries and Configuration Files

Command: find / -perm -2 -type f 2>/dev/null | xargs ls -l

Purpose: To find world-writable files.

Command: find / -perm -u=s -type f 2>/dev/null | xargs ls -l Purpose: To find files with SUID permission.

Command: grep "/bin/bash" /etc/passwd | cut -d':' -f1-4,6,7 Purpose: To identify users with bash shell access.

Selecting Privilege Escalation Command

Command: echo "root2:5ZOYXRFHVZ7OY::0:0:root:/root:/bin/bash" >> /etc/passwd Purpose: To create a new root user entry in the passwd file.

root2: Username.

5ZOYXRFHVZ7OY: Password hash.

::0:0: User and group ID (root).

/root: Home directory.

/bin/bash: Default shell.

Executing the Privilege Escalation Command

Result: Creation of a new root user root2 with a specified password. Remediation Steps Post-Exploitation Remove SUID Bit from cp:

Command: chmod u-s /bin/cp

Purpose: Removing the SUID bit from cp to prevent misuse.

Make Backup Script Not World-Writable:

Command: chmod o-w /path/to/backup/script

Purpose: Ensuring backup script is not writable by all users to prevent unauthorized modifications. Execution and Verification Verifying Hydra Attack:

Run the Hydra command and monitor for successful login attempts.

Verifying Privilege Escalation:

After appending the new root user to the passwd file, attempt to switch user to root2 and check root privileges.

Implementing Remediation:

Apply the remediation commands to secure the system and verify the changes have been implemented.

By following these detailed steps, one can replicate the simulation and ensure a thorough understanding of both the exploitation and the necessary remediations.

QUESTION NO: 3

[Attacks and Exploits]

Which of the following technologies is most likely used with badge cloning? (Select two).

A. NFC

B. RFID

D. Modbus

E. Zigbee

F. CAN bus

ANSWER: A B

Explanation:

Badge cloning is basically copying the data stored on an access badge so it can be replayed by another card or device. The two technologies you'll see most often in real-world access badges are RFID and NFC.

RFID is the big umbrella term for radio-based tags/cards used in lots of door systems (especially older “prox” badges). If an attacker can read the badge’s identifier (and the system doesn’t use strong encryption), they can often write that same data onto a compatible blank card.

NFC is closely related (it’s essentially a short-range form of HF RFID) and is common in newer badges and phones used for tap-to-enter. Because NFC is designed for close-range communication, attackers typically use an NFC reader/writer held near the badge to capture and duplicate what it stores.

Bluetooth, Modbus, Zigbee, and CAN bus don’t really fit the classic “clone a physical access badge” scenario—they’re used for different kinds of device communications, not standard badge credential tech.

References: https://en.wikipedia.org/wiki/Radio-frequency_identification and https://en.wikipedia.org/wiki/Near-field_communication

QUESTION NO: 4 - (DRAG DROP)

DRAG DROP

[Tools and Code Analysis]

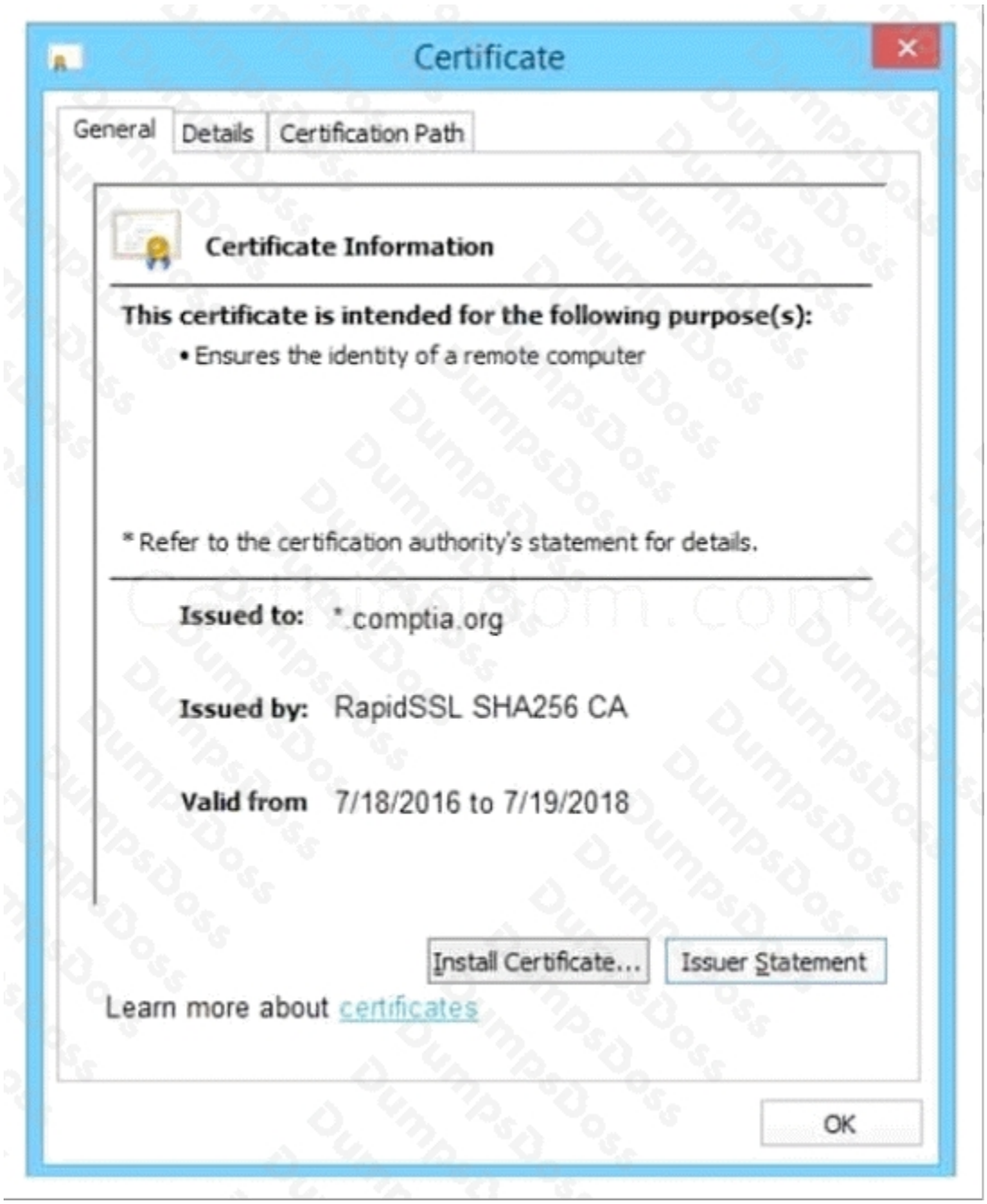
You are a penetration tester reviewing a clients website through a web browser.

INSTRUCTIONS

Review all components of the website through the browser to determine if vulnerabilities are present.

Remediate ONLY the highest vulnerability from either the certificate, source, or cookies.

If at any time you would like to bring back the initial state of the simulation, please click the Reset All button.


Certificate
General
Details
Certification Path
Certificate Information
This certificate is intended for the following purpose(s):
• Ensures the identity of a remote computer
* Refer to the certification authority's statement for details.
Issued to: *.comptia.org
Issued by: RapidSSL SHA256 CA
Valid from 7/18/2016 to 7/19/2018
Install Certificate...
Issuer Statement
Learn more about certificates
OK

Secure System

https://comptia.org/login.aspx#viewsource

```
<html>
<head>
<title>Secure Login </title>
</head>
<body>
<meta
content="c2RmZGZnaHNzZmtqbGdoc2Rma2pnaGRzZmpoZGZvaW2aGRmc29pYmp3ZXindWvdm9pb2hzZGd1aWJoaGR1ZmZpZ2hzZDtpYmhqZHNmc291Ymdoc3d5ZGi1Z2Zi
bnNkbGtqO2Job3VpYXNpZGZubXM7bG8kZmliaHZsb3NhZGJua2N4dnZ1aWdia3NqYWVqa2JmbGI1Y3Z2Z2JobGFzZwJmaXVkZGZidmxiamFmbGhkc3VmZyBuc2pyZ2hzZHVmaG
d1d3NmZ2hqZHNmZmJ1c2hmdWRzZmZoZ3U3cndweWhmamRzZmZ2bnVzZm53cnVMYnZ1ZXJ2==" name="csrt-token"/>
<select> <script>
document.write("<OPTION value=1>"+document.location.href.substring(document.location.href.indexOf("f=")+16)+ "</OPTION>");
</script></select>
<div align="center">
<form action="<c:url value='main.do'/>"method="post">
<div style="margin-top:200px;margin-bottom:10px;">
<span style="width:500px;color:blue;font-size:30px;font-weight:bold;border-bottom:1 px solid blue;">Comptia Secure System Login</span>
</div>
<div style="margin-bottom:5px;">
<span style="width:100px;">Name</span>
<input style="width:150px;"type="text" name="name" id="name" value="">
<!-- input style="width:150px;"type="text" name="name" id="name" value="admin"-->
</div>
<div><span style="width:100px;">Password: </span><input style="width:150px;" type="password" name="Password" id="password" value="">
<!--div><span style="width:100px;">Password: </span><input style="width:150px;" type="password" name="Password" id="password" value="password" -->
```

Secure System

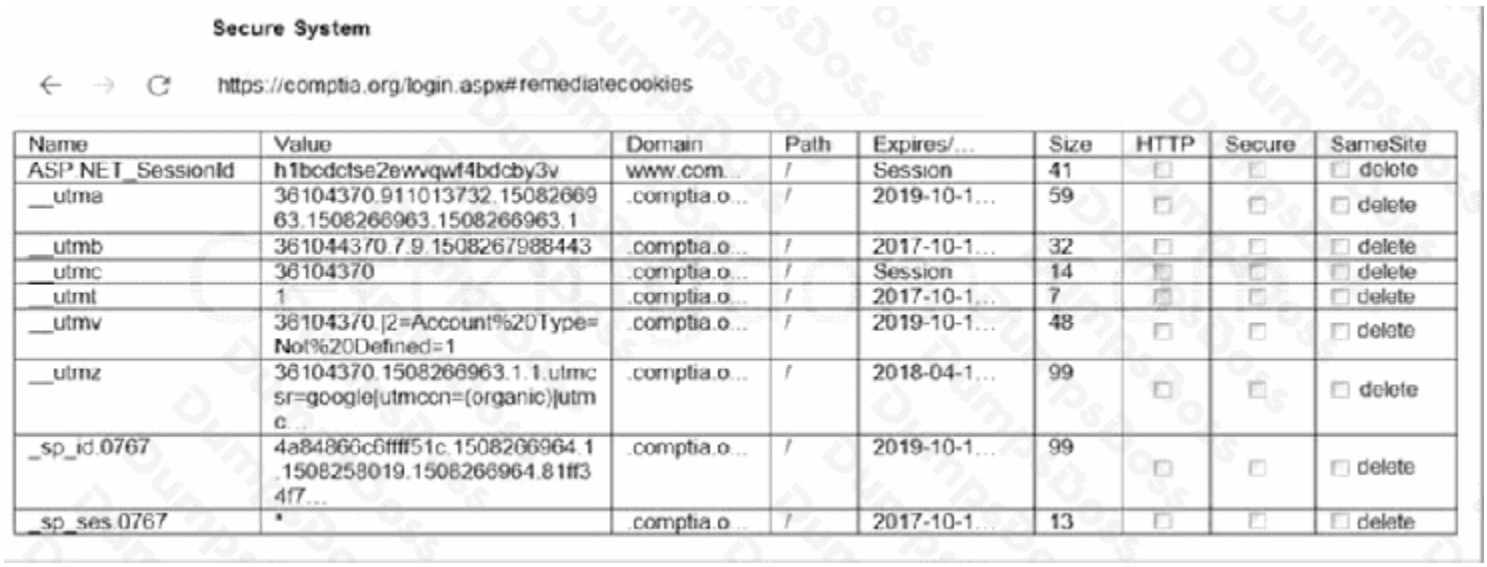
https://comptia.org/login.aspx#viewcookies

Name	Value	Domain	Path	Expires/...	Size	HTTP	Secure	SameSite
ASP.NET_SessionId	h1bcdctse2ewvqwf4bdcby3v	www.com...	/	Session	41			
__utma	36104370.911013732.1508266963.1508266963.1	.comptia.o...	/	2019-10-1...	59			
__utmb	361044370.7.9.1508267988443	.comptia.o...	/	2017-10-1...	32			
__utmc	36104370	.comptia.o...	/	Session	14			
__utmt	1	.comptia.o...	/	2017-10-1...	7			
__utmv	36104370.|2=Account%20Type=Not%20Defined=1	.comptia.o...	/	2019-10-1...	48			
__utmz	36104370.1508266963.1.1.utmcsr=google|utmccn=(organic)|utmc...	.comptia.o...	/	2018-04-1...	99			
_sp_id.0767	4a84866c6ffff51c.1508266964.1.1508258019.1508266964.81ff34f7...	.comptia.o...	/	2019-10-1...	99			
_sp_ses.0767	*	.comptia.o...	/	2017-10-1...	13			

Secure System

https://comptia.org/login.aspx#remediatesource

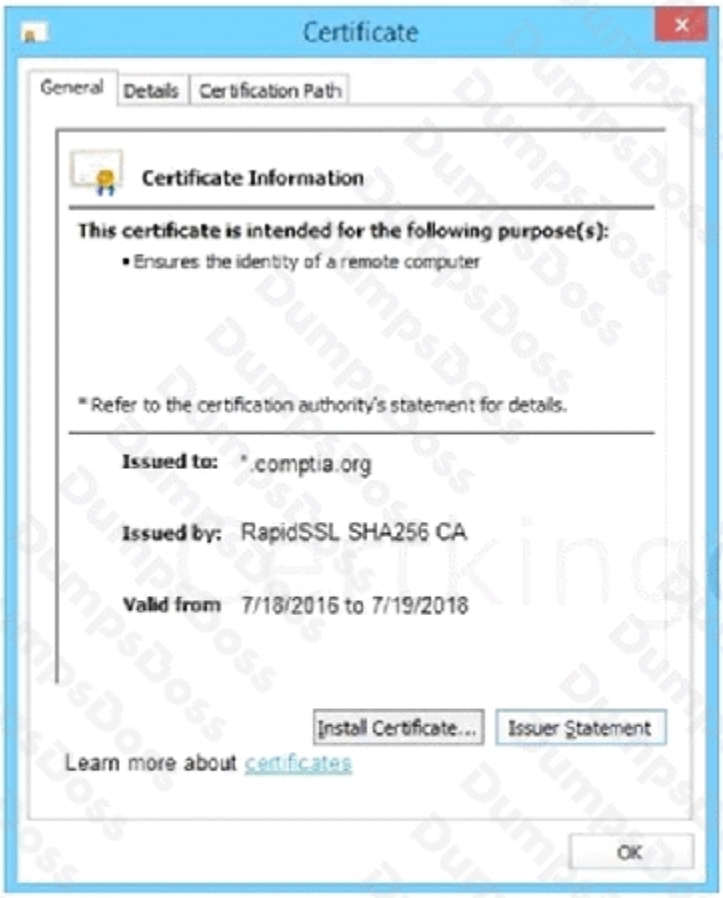
```
1  <html>
2  <head>
3  <title>Secure Login </title>
4  </head>
5  <body>
6  <meta
7  content="c2RmZGZnaHNzZmtqbGdoc2Rma2pnaGRzZmpoZGZvaW2aGRmc29pYmp3ZXindWvdm9pb2hzZGd1aWJoaGR1ZmZpZ2hzZDtpYmhqZHNmc291Ymdoc3d5ZGi
8  bnNkbGtqO2Job3VpYXNpZGZubXM7bGtkZmlaHZsb3NhZGJua2N4dnZ1aWdia3NqYWVqa2JmbGI1Y3Z2Z2JobGFzZwJmaXVkZGZidmxiamFmbGhkc3VmZyBuc2pyZ2hz
9  d1d3NmZ2hqZHNmZmJ1c2hmdWRzZmZoZ3U3cndweWhmamRzZmZ2bnVzZm53cnVMYnZ1ZXJ2==" name="csrt-token"/>
10 <select><script>
11 document.write("<OPTION value=1>"+document.location.href.substring(document.locaton.href.indexOf("f=")+16)+ "</OPTION>");
12 </script></select>
13 <div align="center">
14 <form action="<c:url value='main.do'/>"method="post">
15 <div style="margin-top:200px;margin-bottom:10px;">
16 <span style="width:500px;color:blue;font-size:30px;font-weight:bold;border-bottom:1 px solid blue;">Comptia Secure System Login</span>
17 </div>
18 <div style="margin-bottom:5px;">
19 <span style="width:100px;">Name</span>
20 <input style="width:150px;"type="text" name="name" id="name" value="">
21 <!-- input style="width:150px;"type="text" name="name" id="name" value="admin"-->
22 </div>
23 <div><span style="width:100px;">Password: </span><input style="width:150px;" type="password" name="Password" id="password" value="">
24 <!--div><span style="width:100px;">Password: </span><input style="width:150px;" type="password" name="Password" id="password" value="password" -->
```



Secure System

https://comptia.org/login.aspx#remediatecookies

Name	Value	Domain	Path	Expires/...	Size	HTTP	Secure	SameSite
ASP.NET_SessionId	h1bcdctse2ewvqwf4bdcby3v	www.com...	/	Session	41	☐	☐	☐ delete
__utma	36104370.911013732.15082669 63.1508266963.1508266963.1	.comptia.o...	/	2019-10-1...	59	☐	☐	☐ delete
__utmb	361044370.7.9.1508267988443	.comptia.o...	/	2017-10-1...	32	☐	☐	☐ delete
__utmc	36104370	.comptia.o...	/	Session	14	☐	☐	☐ delete
__utmt	1	.comptia.o...	/	2017-10-1...	7	☐	☐	☐ delete
__utmv	36104370.|2=Account%20Type= Not%20Defined=1	.comptia.o...	/	2019-10-1...	48	☐	☐	☐ delete
__utmz	36104370.1508266963.1.1.utmc sr=google|utmccn=(organic)|utm c...	.comptia.o...	/	2018-04-1...	99	☐	☐	☐ delete
_sp_id.0767	4a84866c6ffff51c.1508266964.1 .1508258019.1508266964.81ff3 4f7...	.comptia.o...	/	2019-10-1...	99	☐	☐	☐ delete
_sp_ses.0767	*	.comptia.o...	/	2017-10-1...	13	☐	☐	☐ delete



Drag and Drop Options:

- Remove certificate from server
- Generate a Certificate Signing Request
- Submit CSR to the CA
- Install re-issued certificate on the server

Step 1

Step 2

Step 3

Step 4

ANSWER:

Explanation:

The clearest “highest vulnerability” shown in the browser artifacts is the TLS certificate problem. In the certificate window, the site’s certificate is valid only from 7/18/2016 to 7/19/2018, meaning it’s expired. An expired certificate is a high-impact issue because users will get certificate warnings and may be trained to click through them, and it undermines the authenticity guarantees TLS is supposed to provide. In a real engagement, this would be treated as a priority fix compared to softer findings like “cookie flags not set” (important, but typically not as immediately trust-breaking as an expired cert) or suspicious commented-out values in HTML (bad practice, but not always directly exploitable without additional context).

To remediate an expired certificate, the correct operational flow is: create a new key pair/CSR, submit the CSR to a trusted CA, obtain the re-issued certificate, and then install it on the server so it replaces the old one. If the simulation forces an explicit “remove certificate from server” step, that removal should happen before installing the new certificate (or be considered part of the replacement process). Placing “remove certificate” after “install re-issued certificate” is backwards because the goal is to keep the service available and simply replace the old certificate with the new one during the installation step.

References: NIST’s guidance on TLS and certificate management emphasizes maintaining valid certificates and proper lifecycle handling (https://csrc.nist.gov/publications/detail/sp/800-52/rev-2/final). For practical CA/CSR lifecycle context, see Let’s Encrypt’s explanation of CSRs and issuance (https://letsencrypt.org/docs/csr/).

QUESTION NO: 5

[Attacks and Exploits]

A penetration tester gains initial access to a target system by exploiting a recent RCE vulnerability. The patch for the vulnerability will be deployed at the end of the week. Which of the following utilities would allow the tester to reenter the system remotely after the patch has been deployed? (Select two).

A. schtasks.exe

B. rundll.exe

C. cmd.exe

D. chgusr.exe

E. sc.exe

F. netsh.exe

ANSWER: A E

Explanation:

If the RCE gets patched, the original way in is gone, so the tester needs persistence—something that will still run even after updates and reboots. On Windows, two very common “living off the land” options are scheduled tasks and services.

schtasks.exe can create a scheduled task that runs a payload on a timer or at logon (often as SYSTEM). Once it’s in place, the patch doesn’t matter, because the system itself will keep launching the task and the tester can regain access through that backdoor. Microsoft’s overview is here: https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/schtasks

sc.exe can create or modify a Windows service to start automatically. Services are a classic persistence method because they can run in the background and start at boot, again independent of the original vulnerability. Microsoft’s documentation is here: https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/sc

The other options don’t really “stick” by themselves: **cmd.exe** is just a shell, **netsh.exe** is for networking, **chgusr.exe** is unrelated, and **rundll.exe** can execute DLLs but isn’t a persistence mechanism unless paired with something like a service/task or registry run key.

QUESTION NO: 6

[Tools and Code Analysis]

In a file stored in an unprotected source code repository, a penetration tester discovers the following line of code:

sshpass -p donotchange ssh admin@192.168.6.14

Which of the following should the tester attempt to do next to take advantage of this information? (Select two).

A. Use Nmap to identify all the SSH systems active on the network.

B. Take a screen capture of the source code repository for documentation purposes.

C. Investigate to find whether other files containing embedded passwords are in the code repository.

D. Confirm whether the server 192.168.6.14 is up by sending ICMP probes.

E. Run a password-spraying attack with Hydra against all the SSH servers.

F. Use an external exploit through Metasploit to compromise host 192.168.6.14.

ANSWER: B C

Explanation:

That line is basically a gift: it shows a hard-coded SSH password and the exact target host and username. The smartest "next move" is to capture solid evidence of what you found. A quick screenshot (or other proof like a copy of the file path and commit hash) makes reporting easier and helps the client reproduce and fix the issue later.

After that, don't assume it's the only secret in the repo. If someone hard-coded one password, there's a decent chance there are more—API keys, tokens, other server creds, etc. So the next practical step is to search the repository for additional embedded secrets using simple grep searches or secret-scanning tools. This often leads to more access and a more complete finding.

Options like broad Nmap scanning, ICMP checks, password spraying, or jumping straight to Metasploit are noisier and not really "using" the repo intel first. The clean win here is: document the credential leak and mine the repo for more secrets. References: https://owasp.org/www-project-top-ten/ (A02: Cryptographic Failures / secret exposure) and https://github.com/trufflesecurity/trufflehog (common tool for finding leaked secrets).

QUESTION NO: 7

During an assessment, a penetration tester sends the following request:

POST /services/v1/users/create HTT

P.1

Host: target-application.com

Content-Type: application/json

Content-Length: [dynamic]

Authorization: Bearer (FUZZ)

Which of the following attacks is the penetration tester performing?

A. Directory traversal

B. API abuse

C. Server-side request forgery

D. Privilege escalation

ANSWER: B

Explanation:

The big clue here is **Authorization: Bearer (FUZZ)**. "FUZZ" usually means the tester is feeding lots of different values into that spot (random strings, malformed tokens, expired tokens, tokens with tweaked claims, etc.) to see how the API reacts. In

plain terms, they're probing the API's authentication/authorization handling to find weak checks, token validation bugs, or logic mistakes.

That lines up best with **API abuse**, because the tester is intentionally misusing the API interface (specifically the auth header) to look for ways to bypass controls or trigger unexpected behavior. This isn't directory traversal (no file paths like `../`), and it doesn't look like SSRF (nothing is trying to make the server fetch a URL). Privilege escalation could be a possible end result if a bad token is accepted, but the action shown is the fuzzing/testing step—so "API abuse" is the most accurate label.

References: https://owasp.org/www-project-api-security/ and https://owasp.org/www-community/Fuzzing

QUESTION NO: 8

[Information Gathering and Vulnerability Scanning]

A penetration tester identifies the following open ports during a network enumeration scan:

PORT STATE SERVICE

22/tcp open ssh

80/tcp open http

111/tcp open rpcbind

443/tcp open https

27017/tcp open mongodb

50123/tcp open ms-rpc

Which of the following commands did the tester use to get this output?

A. nmap -Pn -A 10.10.10.10

B. nmap -sV 10.10.10.10

C. nmap -Pn -w 10.10.10.10

D. nmap -sV -Pn -p- 10.10.10.10

ANSWER: D

Explanation:

The big clue in the output is that it includes a high, non-default port (50123/tcp). A normal Nmap scan without extra flags usually checks only the "top" common ports, and it would be easy to miss something random like 50123.

That's why the command needs `-p-`, which tells Nmap to scan all 65,535 TCP ports. On top of that, the output shows service names (like mongodb and ms-rpc), which points to service detection being enabled with `-sV`.

Finally, `-Pn` makes sense in real pentests because it skips host discovery (ping checks). If ICMP is blocked, Nmap might otherwise assume the host is down and not scan it properly. Putting those together, `nmap -sV -Pn -p- 10.10.10.10` best matches the kind of results shown.

Reference: https://nmap.org/book/man-briefoptions.html and https://nmap.org/book/man-port-specification.html

QUESTION NO: 9

[Attacks and Exploits]

A penetration tester finds that an application responds with the contents of the /etc/passwd file when the following payload is sent:

]>

&foo;

Which of the following should the tester recommend in the report to best prevent this type of vulnerability?

A. Drop all excessive file permissions with chmod o-rwx

B. Ensure the requests application access logs are reviewed frequently

C. Disable the use of external entities

D. Implement a WAF to filter all incoming requests

ANSWER: C

Explanation:

This payload is a classic XXE (XML External Entity) attack. The app is parsing XML and happily resolving an external entity that points to a local file (file:///etc/passwd). That's why the response comes back with the file contents—your XML parser is basically being tricked into reading server-side files for the attacker.

The best fix is to stop the parser from resolving external entities (and ideally disable DTDs entirely if the app doesn't truly need them). Once external entity resolution is turned off, the &foo; reference won't get expanded, so the attacker can't use XML to pull local files or hit internal network resources. This is the direct, root-cause mitigation.

A WAF can sometimes catch obvious XXE payloads, but it's not as reliable as fixing the parser configuration, and attackers can often bypass filtering. Changing file permissions on /etc/passwd isn't realistic (and doesn't solve the underlying issue), and reviewing logs is detection, not prevention.

References: https://owasp.org/www-community/vulnerabilities/XML_External_Entity_(XXE)_Processing and https://cheatsheetseries.owasp.org/cheatsheets/XML_External_Entity_Prevention_Cheat_Sheet.html

QUESTION NO: 10

A penetration tester presents the following findings to stakeholders:

Control | Number of findings | Risk | Notes

Encryption | 1 | Low | Weak algorithm noted

Patching | 8 | Medium | Unsupported systems

System hardening | 2 | Low | Baseline drift observed

Secure SDLC | 10 | High | Libraries have vulnerabilities

Password policy | 0 | Low | No exceptions noted

Based on the findings, which of the following recommendations should the tester make? (Select two).

A. Develop a secure encryption algorithm.

B. Deploy an asset management system.

C. Write an SDLC policy.

D. Implement an SCA tool.

E. Obtain the latest library version.

F. Patch the libraries.

ANSWER: D E

Explanation:

The biggest red flag here is the **Secure SDLC** category: it has the most findings and it's marked **High** risk because the app is pulling in vulnerable libraries. That's a classic "dependency risk" problem, so the best recommendations are the ones that help you *find* and *fix* bad third-party components quickly and consistently.

Implementing an SCA tool is a solid move because SCA (Software Composition Analysis) scans your software dependencies (open-source and third-party) and flags known vulnerable versions. It also helps teams track what's in use and stay on top of updates over time. That directly targets the Secure SDLC issue. See: https://owasp.org/www-community/Component_Analysis

Obtaining the latest library version is the practical "do it now" fix. If the libraries are vulnerable, updating to patched versions is often the fastest way to reduce risk (assuming compatibility testing). This aligns with OWASP guidance around keeping components updated. See: https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/

QUESTION NO: 11

[Attacks and Exploits]

Which of the following is a term used to describe a situation in which a penetration tester bypasses physical access controls and gains access to a facility by entering at the same time as an employee?

A. Badge cloning

B. Shoulder surfing

C. Tailgating

D. Site survey

ANSWER: C

Explanation:

The situation described is classic **tailgating**. That's when someone who isn't authorized gets through a secured door simply by walking in right behind an employee who *is* authorized—often because the employee is being polite, distracted, or assumes the person belongs there.

This is different from things like badge cloning (copying an access card) or shoulder surfing (watching someone type a PIN/password). A site survey is more like recon—looking around and gathering info about a location—rather than actually slipping inside.

In real physical pentests, tailgating is a common technique because it tests the "human" side of security. Even with strong locks and badge readers, one person holding the door can undo the whole control. Good defenses include mantraps, turnstiles, security guards, and training employees to challenge or report unknown visitors.

References: https://en.wikipedia.org/wiki/Tailgating_(security) and https://www.cisa.gov/news-events/news/avoid-tailgating-and-piggybacking

QUESTION NO: 12

[Attacks and Exploits]

A penetration tester is conducting a wireless security assessment for a client with 2.4GHz and 5GHz access points. The tester places a wireless USB dongle in the laptop to start capturing WPA2 handshakes. Which of the following steps should the tester take next?

A. Enable monitoring mode using Aircrack-ng.

B. Use Kismet to automatically place the wireless dongle in monitor mode and collect handshakes.

C. Run KARMA to break the password.

D. Research WiGL
E.net for potential nearby client access points.

ANSWER: A

Explanation:

To capture a WPA2 handshake, the adapter has to be able to "listen to everything" on the air, not just traffic meant for its own MAC address. That's exactly what monitor mode is for. So right after plugging in the USB Wi-Fi dongle, the next practical step is putting that interface into monitor mode (commonly with tools from the aircrack-ng suite like `airmon-ng`). Once it's in monitor mode, you can use something like `airodump-ng` to watch for the handshake and save the capture.

Option B sounds tempting because Kismet can use monitor mode, but it's not the "next step" you rely on universally—monitor mode is the requirement, and Kismet isn't guaranteed to automatically set it up the way you need for your workflow. KARMA (C) is a rogue AP/evil twin style attack and doesn't "break the password" directly. WiGLE (D) is for mapping and recon, not for collecting handshakes during an on-site assessment.

References: https://www.aircrack-ng.org/documentation.html and https://www.kismetwireless.net/docs/readme/

QUESTION NO: 13

During a testing engagement, a penetration tester compromises a host and locates data for exfiltration. Which of the following are the best options to move the data without triggering a data loss prevention tool? (Select two).

A. Move the data using a USB flash drive.

B. Compress and encrypt the data.

C. Rename the file name extensions.

D. Use FTP for exfiltration.

E. Encode the data as Base64.

F. Send the data to a commonly trusted service.

ANSWER: B E

Explanation:

DLP tools usually catch exfiltration by inspecting content (patterns like SSNs, card numbers, keywords) and watching for suspicious transfers. If you compress and encrypt the data, you're making it both smaller and unreadable to content inspection. A lot of DLP solutions can't "look inside" encrypted archives unless they have the keys, so the sensitive patterns they rely on simply aren't visible anymore.

Base64 encoding is another common way to sneak past basic content checks. It turns raw data into ASCII text, which can slip through controls that are mostly tuned for file types or obvious sensitive strings. It's not as strong as encryption, but it's a practical obfuscation method that can reduce signature-based hits, especially in less mature DLP setups.

The other choices are weaker: renaming extensions doesn't change the real content, FTP is noisy and often monitored, USB exfiltration is risky in managed environments, and "trusted services" are frequently logged and scanned by CASB/DLP integrations.

References: https://attack.mitre.org/techniques/T1027/ (obfuscation) and https://attack.mitre.org/techniques/T1560/ (archive/compress collected data).

QUESTION NO: 14

[Tools and Code Analysis]

A penetration tester compromises a Windows OS endpoint that is joined to an Active Directory local environment. Which of the following tools should the tester use to manipulate authentication mechanisms to move laterally in the network?

A. Rubeus

B. WinPEAS

C. NTLMRelayX

D. Impacket

ANSWER: A

Explanation:

In an Active Directory environment, a lot of lateral movement comes down to messing with authentication, and that usually means Kerberos. Rubeus is built specifically for Kerberos abuse, so it's the most direct fit when you want to do things like extract or request tickets, perform pass-the-ticket, or run Kerberoasting to get credentials you can reuse across the domain.

WinPEAS is great, but it's more of a "what can I escalate locally on this box?" helper. It's not the go-to for manipulating AD authentication flows. NTLMRelayX can be powerful when you're relaying NTLM auth, but the question is pointing at AD lateral movement by manipulating authentication mechanisms in general—Kerberos is the big one there. Impacket is a broader toolkit with lots of useful scripts, but it's not as purpose-built for Kerberos ticket games as Rubeus is.

References: https://github.com/GhostPack/Rubeus and https://attack.mitre.org/techniques/T1558/

QUESTION NO: 15

[Reporting and Communication]

Which of the following are valid reasons for including base, temporal, and environmental CVSS metrics in the findings section of a penetration testing report? (Select two).

A. Providing details on how to remediate vulnerabilities

B. Helping to prioritize remediation based on threat context

C. Including links to the proof-of-concept exploit itself

D. Providing information on attack complexity and vector

E. Prioritizing compliance information needed for an audit

F. Adding risk levels to each asset

ANSWER: B D

Explanation:

Including CVSS base, temporal, and environmental metrics helps the reader understand both the "what" and the "so what" of a vulnerability. The base metrics explain the built-in technical severity—things like whether it's exploitable over the network, how hard it is to pull off, and what kind of impact it can have. That's why "attack complexity and vector" is a solid reason to include CVSS in the findings.

It also helps teams prioritize fixes in a way that matches real life. Temporal metrics reflect how the situation changes over time (for example, if a working exploit is now public), and environmental metrics let the organization tune the score based on their own setup and business impact. That combination makes it much easier to decide what to patch first instead of treating every finding like it's equally urgent.

CVSS isn't meant to provide step-by-step remediation, PoC links, audit checklists, or asset classification. It's mainly a consistent scoring language for vulnerability severity and context. References: https://www.first.org/cvss/ and https://nvd.nist.gov/vuln-metrics/cvss