? Easy Level (10 Questions)

Q1. Which Python module is primarily used to create client–server communication?
A. `os`
B. `socket`
C. `sys`
D. `threading`

Q2. In socket programming, which component initiates the connection?
A. Server
B. Client
C. Router
D. Firewall

Q3. Which function prepares a server socket to accept connections?
A. `connect()`
B. `bind()`
C. `listen()`
D. `send()`

Q4. What is the main purpose of monitoring client activity on a server?
A. Improve UI
B. Increase bandwidth
C. Detect abnormal behavior
D. Store backups

Q5. Which tool category helps observe runtime behavior of programs?
A. Compiler
B. Debugger
C. Linker
D. Assembler

Q6. What does debugging primarily aim to do?
A. Write exploits
B. Optimize networks

C. Identify and fix errors
D. Encrypt traffic

Q7. Which socket type provides reliable, ordered data delivery?
A. UDP
B. RAW
C. TCP
D. ICMP

Q8. Which programming language is commonly used to write plugins?
A. C
B. Java
C. Python
D. Assembly

Q9. Which activity is part of exploit analysis rather than exploit creation?
A. Payload crafting
B. Vulnerability identification
C. Buffer overflow writing
D. Shellcode injection

Q10. Which Python feature helps automate repetitive analysis tasks?
A. Decorators
B. Scripts
C. Threads
D. Classes

Medium Level (15 Questions)

Q11. Which socket function returns a new socket for client communication?
A. `connect()`
B. `accept()`
C. `listen()`
D. `recv()`

Q12. Which data is most useful when monitoring client behavior?
A. Screen resolution
B. Packet timestamps
C. Keyboard layout
D. CPU brand

Q13. Which Python library helps log and analyze runtime events?
A. `logging`
B. `datetime`
C. `hashlib`
D. `random`

Q14. What is a plugin primarily designed to do?
A. Replace core software
B. Extend functionality
C. Reduce memory usage
D. Disable services

Q15. Which exploit development stage focuses on understanding root cause?
A. Delivery
B. Exploitation
C. Vulnerability analysis
D. Privilege escalation

Q16. Which debugging technique pauses execution at specific points?
A. Tracing
B. Breakpoints
C. Profiling
D. Logging

Q17. Which socket method receives data from a client?
A. `send()`
B. `recv()`
C. `bind()`
D. `accept()`

Q18. Which automation benefit is most relevant in exploit analysis?
A. Reduced bandwidth
B. Faster repeatability
C. Improved UI
D. Increased compilation speed

Q19. Which programming concept is essential for writing modular plugins?
A. Global variables
B. Hard coding
C. Interfaces / APIs
D. Inline assembly

Q20. Which server-side risk can be detected by monitoring socket traffic?
A. Disk fragmentation
B. Unauthorized access attempts
C. CPU overheating
D. Power failure

Q21. Which debugging output helps track function execution order?
A. Memory dump
B. Stack trace
C. Binary diff
D. Packet capture

Q22. Which exploit vector commonly uses sockets?
A. Local file inclusion
B. Network-based attacks
C. Hardware faults
D. Power analysis

Q23. Which automation approach reduces manual exploit testing effort?
A. Static typing
B. Scripted testing

C. Manual debugging
D. GUI analysis

Q24. Which plugin design improves maintainability?
A. Monolithic code
B. Tight coupling
C. Modular architecture
D. Inline dependencies

Q25. Which debugging tool category inspects memory at runtime?
A. Network analyzer
B. Memory debugger
C. Text editor
D. Compiler

Hard Level (15 Questions)

Q26. Which server-side mechanism best enables detailed client activity monitoring?
A. NAT
B. Session logging
C. Load balancing
D. DNS caching

Q27. Which exploit analysis step validates exploit reliability?
A. Fingerprinting
B. Fuzzing
C. Post-exploitation
D. Obfuscation

Q28. Which Python construct best supports extensible plugin frameworks?
A. Lambda functions
B. Abstract base classes
C. Global variables
D. Inline imports

Q29. Which socket-level metric helps detect abnormal client behavior?
A. Packet entropy
B. Latency patterns
C. Screen DPI
D. Keyboard scan codes

Q30. Which debugging issue occurs when variable values are optimized out?
A. Deadlock
B. Race condition
C. Heisenbug
D. Stack overflow

Q31. Which exploit automation component orchestrates multiple analysis steps?
A. Compiler
B. Controller script
C. Payload
D. Shell

Q32. Which technique helps isolate faulty exploit logic?
A. Binary packing
B. Step-by-step debugging
C. Encryption
D. Compression

Q33. Which socket programming flaw can enable exploitation?
A. Strong authentication
B. Proper input validation
C. Lack of input sanitization
D. Encrypted communication

Q34. Which debugging method captures function calls without stopping execution?
A. Breakpoints
B. Tracing
C. Core dumps
D. Snapshots

Q35. Which exploit development goal focuses on long-term reliability?
A. Speed
B. Portability
C. Stealth
D. Persistence

Q36. Which plugin capability supports automated exploit analysis?
A. Hardcoded values
B. Event hooks
C. Static binaries
D. Manual triggers

Q37. Which socket behavior may indicate a denial-of-service attempt?
A. Balanced traffic
B. Repeated rapid connections
C. Normal handshake
D. Encrypted packets

Q38. Which debugging artifact helps analyze crashes post-execution?
A. Log file
B. Stack trace
C. Core dump
D. Packet log

Q39. Which automation challenge arises in exploit analysis?
A. Over-documentation
B. Environment variability
C. Excess bandwidth
D. UI consistency

Q40. Which secure practice mitigates exploitation through sockets?
A. Default credentials
B. Input validation and rate limiting
C. Open ports
D. Anonymous access