**«ПОДТВЕРЖДАЮ»**

**Заведующей кафедрой**

**«Криптология»**

**\_\_\_\_\_\_\_\_\_\_\_\_ Худойкулов З.Т.**

**«\_\_\_\_»\_\_\_\_\_\_\_\_2024 год**

**Вопросы для итогового контроля**

**по дисциплине «Безопасность программных средств»**

1. Проблемы безопасной разработки программного обеспечения
2. Основные проблемы при разработки программного обеспечения
3. Проблемы разработки (ошибка, дефект, сбой, неисправность)
4. Security Development Lifecycle (жизненный цикл безопасной разработки)
5. Методы обеспечения конфиденциальности и целостности
6. Методы обеспечения и контроля доступности
7. Вредоносные программы и методы защиты
8. Жизненный цикл разработки программного обеспечения (SDLC) – этапы
9. Жизненный цикл разработки программного обеспечения (SDLC) - связь (Communication)
10. Жизненный цикл разработки программного обеспечения (SDLC) – сбор требований (Requirement gathering)
11. Жизненный цикл разработки программного обеспечения (SDLC) – экономически-техническое обоснование (Feasibility Study)
12. Жизненный цикл разработки программного обеспечения (SDLC) – системный анализ (System Analysis)
13. Жизненный цикл разработки программного обеспечения (SDLC) – проектирование программного обеспечения (Software Design)
14. Жизненный цикл разработки программного обеспечения (SDLC) – коддинг (Coding)
15. Жизненный цикл разработки программного обеспечения (SDLC) – тестирование (Testing)
16. Жизненный цикл разработки программного обеспечения (SDLC) – интеграция (Integration)
17. Жизненный цикл разработки программного обеспечения (SDLC) – развертывание (Implementation, deployment)
18. Жизненный цикл разработки программного обеспечения (SDLC) – поддержка (maintenance)
19. Модели жизненного цикла программного обеспечения – каскадная модель
20. Модели жизненного цикла программного обеспечения – итерационная модель
21. Модели жизненного цикла программного обеспечения – спиральная модель
22. Модели жизненного цикла программного обеспечения – инкрементная модель
23. Модели жизненного цикла программного обеспечения – V-модель
24. Требования к программному обеспечению – этапы
25. Требования к программному обеспечению – источники и пути выявления требований
26. Требования к программному обеспечению – характеристики требований к программному обеспечению
27. Спецификация требований программного обеспечения (Software Requirements Specification, SRS)
28. Требования к программному обеспечению – функциональные требования
29. Требования к программному обеспечению – нефункциональные требования
30. Требования к качеству и оценка программного продукта
31. Архитектура и проектирование программного обеспечения
32. Диаграмма потока данных (data flow diagram, dfd)
33. Архитектура программного обеспечения архитектура – труб и фильтров (pipe and filter);
34. Архитектура программного обеспечения архитектура – архитектура управления процессом
35. Архитектура программного обеспечения архитектура – клиент-серверная архитектура
36. Архитектура программного обеспечения архитектура – объектно-ориентированная архитектура
37. Архитектура программного обеспечения архитектура – слоистый стиль
38. Архитектура программного обеспечения архитектура – брокер архитектурный стиль
39. Архитектура программного обеспечения архитектура – архитектурный стиль доски
40. Уязвимости программного обеспечения и их причины
41. Уязвимости OWASP - broken access control
42. Уязвимости OWASP - cryptographic failures
43. Уязвимости OWASP – injection
44. Уязвимости OWASP - insecure design
45. Уязвимости OWASP - security misconfiguration
46. Уязвимости OWASP - vulnerable and outdated components
47. Уязвимости OWASP - identification and authentication failures
48. Уязвимости OWASP - software and data integrity failures
49. Уязвимости OWASP - security logging and monitoring failures
50. Уязвимости OWASP - server side request forgery
51. Языки программирования и функции безопасности
52. Характеристики языков программирования
53. Языки программирования низкого уровня: преимущества
54. Языки программирования высокого уровня: преимущества
55. Составления кода программного обеспечения
56. Утечка информации (Information Flow)
57. Уязвимости программных средств – переполнение буфера
58. Sandboxing (песочница)
59. Уязвимости программных средств – висячий указатель или висячая ссылка (Dandling)
60. Уязвимости программных средств – инъекция кода
61. Уязвимости программных средств – XSS (Cross-Site Scripting) в веб-приложениях
62. Классификация угроз методология STRIDE
63. Уязвимости программных средств – SQL-инъекция
64. Статический анализ кода
65. Уязвимости программных средств – строка неконтролируемого формата (Uncontrolled Format String)
66. Инструменты статического анализа программного обеспечения
67. Уязвимости программных средств – внедрение команд ОС
68. Развертывание программного обеспечения (Software deployment)
69. Уязвимости программных средств – Time-of-check-to-time-of-use
70. Требования безопасности к программному обеспечению
71. Тестирования безопасности программных средств
72. Методы тестирования безопасности программных средств
73. Инструменты статического анализа программного обеспечения
74. Реверс инжиниринг и инструменты
75. Подготовка к внедрению программного обеспечения
76. Что такое статический анализ программного средства?
77. Как работает статический анализ программного средства?
78. Что такое статический и динамический анализ программного средства?
79. Объясните требование по кодированию MISRA.
80. Объясните стандарт ISO 26262: ASIL.
81. Тестирование безопасности программного обеспечения, его виды.
82. Методы тестирования: тестирование «черного ящика».
83. Методы тестирования: тестирование «белого ящика».
84. Методы тестирования: тестирование «серого ящика».
85. Анализ псевдокода (определение результатов путем анализа различных заданных кодов)   
    *begin  
    numeric nNum1,nNum2, i, nSum  
    display "ENTER THE FIRST NUMBER : "  
    accept nNum1 // nNum1>5  
    display "ENTER THE SECOND NUMBER : "  
    accept nNum2 // nNum2>5*

*nSum=0*

*FOR i=2 TO nNum2 – 1*

*IF nNum2 MOD i ==0*

*nSum=nSum+1*

*END IF*

*END FOR*

*IF nSum==0*

*display " nNum1 " ELSE display " nNum2 "*

*END IF*

*END*

1. Анализ псевдокода (определение результатов путем анализа различных заданных кодов)   
   *begin  
   numeric nNum1,nNum2, i, nSum, j  
   display "ENTER THE FIRST NUMBER : "  
   accept nNum1 // nNum1> 5  
   display "ENTER THE SECOND NUMBER : "  
   accept nNum2 // nNum2>5*

*i=4*

*WHILE i< nNum2*

*nSum=0 j=2*

*WHILE j< i*

*IF i MOD j ==0*

*nSum=nSum+1*

*END IF*

*j=j+1*

*END WHILE*

*IF nSum==0 display i, " "*

*END IF*

*i=i+1*

*END WHILE*

*END*

1. Анализ псевдокода (определение результатов путем анализа различных заданных кодов)   
   *begin  
   numeric nNum1, i, max*

*numeric a[20]  
display "ENTER THE FIRST NUMBER : "  
accept nNum1 // nNum1> 5  
 FOR i=0 TO nNum1 – 1*

*accept a[i]*

*END FOR*

*max=a[1]*

*FOR i=0 TO nNum1 – 1*

*IF a[i] >max*

*max=max Else max=a[i]*

*END IF*

*END FOR*

*Output max END*

1. Анализ псевдокода (определение результатов путем анализа различных заданных кодов)   
   *begin  
   numeric nNum1,nNum2, i, nSum  
   display "ENTER THE FIRST NUMBER : "  
   accept nNum1 // nNum1>5  
   display "ENTER THE SECOND NUMBER : "  
   accept nNum2 // nNum2>5*

*nSum=0*

*FOR i=2 TO nNum2 – 1*

*IF nNum2 MOD i ==0*

*nSum=nSum+1*

*END IF*

*END FOR*

*IF nSum==0*

*display " nNum1 " ELSE display " nNum2 "*

*END IF*

*END*

1. Уязвимость программного обеспечения: Переполнение буфера Объяснить с примерами.
2. Уязвимость программного обеспечения: Dandling Indicator объяснить на примерах.
3. Уязвимость программного обеспечения: SQL Injection объяснить с примерами.
4. Уязвимость программного обеспечения: OS Command Injection.
5. Уязвимость программного обеспечения: Целочисленное переполнение буфера объяснить на примерах.
6. Уязвимость программного обеспечения: Time-of-check-to-time-of-use (TOCTOU).
7. Объяснить уязвимости и меры защиты, объявленные OWASP.
8. Объяснить классификация угроз.
9. Объяснить модель угроз и ее основное назначение.
10. Объяснить методология STRIDE.
11. Объяснить безопасность на основе языков программирования и ее возможности.
12. Объяснить безопасность на основе языков программирования: безопасность памяти.

**Составитель:**

**ассистент кафедры, «Криптология»: Хамидов Ш.Ж.**