Teach Me Skills

Кодинг и уязвимости

часть 2

Teach
Me
Skills

Собираемся и отмечаемся

Teach Me Skills

Вопросы по предыдущим темам или ДЗ

Teach Me Skills

Mini-quize по прошлым темам:

1. **Что такое КВОИ?**
2. **Какие основные документы регламентируют деятельность отделов / ЦК?**
3. **Какие IOC первостепенны к сбору?**
4. **Для чего проводится аттестация ЦОК/ЦК?**

Teach Me Skills

Mini-quize по новой теме:

1. **Какие математические операции в программировании вы знаете?**
2. **Какие регулярные выражения вы знаете?**
3. **К чему может привести отсутствие обработчиков ошибок и инструментов проверки кода?**
4. **В чем заключается основная проблема при отсутствии валидации введенных данных?**

Teach Me Skills

План занятия

1. Математические операции и операторы сравнения на примере Python
2. Регулярные выражения
3. Отсутствие шифрования и использование слабых алгоритмов шифрования
4. Ключ шифрования или пароли в исходном коде
5. Слабые и сильные алгоритмы хеширования

Математические операторы

Оператор	Операция
-	Вычитание, также унарный минус
+	Сложение
*	Умножение
/	Деление
%	Остаток от деления
--	Декремент или уменьшение
++	Инкремент или увеличение

Пример кода

```
int a = 7, b = 1, c = 0;
c += a;
a += b;
c = a * b;
c -= a / b;
c--;
```

Операторы сравнения

Teach Me Skills

Оператор	Операция
>	Больше чем
>=	Больше или равно
<	Меньше чем
<=	Меньше или равно
==	Равно
!=	Не равно
&&	И
||	ИЛИ
!	НЕ, отрицание

Пример применения

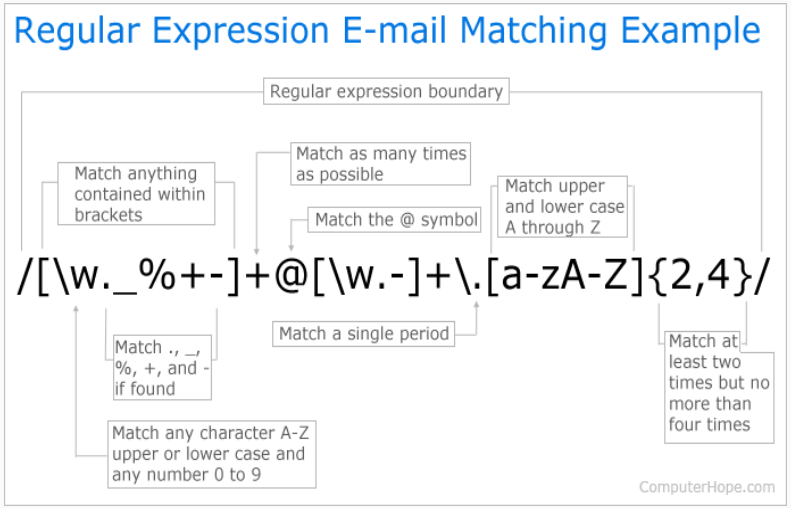
```
int a = 10, b = 5;
bool flag = true, flag2 = false;

if(flag == true || flag != true){

    if((a > 4 && b <= 6) ||
        (a >= 10 && b == 5)){

        cout << "Work!" << endl;
    }
}
```

Регулярные выражения — формальный язык, используемый в компьютерных программах, работающих с текстом, для поиска и осуществления манипуляций с подстроками в тексте, основанный на использовании метасимволов. Для поиска используется строка-образец, состоящая из символов и метасимволов и задающая правило поиска.



В python за регулярные выражения отвечает модуль «re»

Import re

- ***re.match()*** - ищет по заданному шаблону в начале строки
- ***re.search()*** - ищет по всей строке, но возвращает первое найденное совпадение
- ***re.findall()*** - возвращает список всех найденных совпадений
- ***re.compile()*** - собирает регулярное выражение в отдельный объект

Кодовые символы regex

^ - начало строки
$ - конец строки
\b – границы слова
\d – любая цифра
\D – любая не цифра
\s – любой пробельный символ
\S – любой не пробельный символ
\w – любая буква
\W – любой не буквенный символ
[a-z] – диапазон букв(чисел)
[a-zA-Z] – двойной диапазон букв(чисел)
. – любой символ
a|b – Или а или б
[^a-z] – Любая буква(цифра) не из диапазона
A+ - буква а один или более одного раза

\x – hex символ
[\b] – символ backspace
b* – любое количество символов b
b? – не жадная выборка b
b*? –не жадная выборка любого количества b
(?'group'...) – группа
(?P<group>...) – группа golang
(?<group>...) - группа
A{3} – количество повторений A
A{3,} – минимум три повторения A
A{3, 7} – от 3-ех до 7-ми повторений A

Форматная строка

Teach Me Skills

Форматная строка - это строка в программировании, которая определяет, как должны быть отформатированы или представлены определенные значения в текстовом виде

```
int number = 42;
printf("The answer is %d\n", number);

scanf("%d", &number);

scanf(спецификатор, переменная);
printf(строка+спецификатор, переменная);
```

Спецификаторы формата в C и C++, используются в функциях, таких как **printf**, **scanf**.
Они указывают, как интерпретировать аргументы, переданные в функцию, и форматировать вывод или ввод.

1.Спецификаторы для вывода/ввода:

%d - целое число в десятичной системе.
%x, **%X** - целое число в шестнадцатеричной системе.
%u - беззнаковое целое число.
%s - строка.
%f, **%lf** - число с плавающей точкой (float, double)

2.Модификаторы ширины и точности:

%5d - ширина поля вывода в пять символов.
%.2f - два знака после точки для чисел с плавающей точкой.

3. Дополнительные спецификаторы:

%n - записывает количество успешно записанных символов в указанную переменную (например, **printf("%n", &count)**).

```
int num = 11;
printf("%d", num);   // Вывод целого числа: 11

char str[] = "It's me!";
printf("%s", str);   // Вывод строки: It's me!

float pi = 3.14159;
printf("%.2f", pi);  // Вывод числа с двумя знаками после точки: 3.14
```

Алгоритмы хэширования

Сильные алгоритмы:

SHA-256 : SHA-256 является частью семейства алгоритмов SHA-2 и использует 256-битные хеш-значения. Он обеспечивает высокий уровень стойкости к коллизиям и является одним из самых распространенных и безопасных хеш-алгоритмов.

SHA-3 : SHA-3 является последним стандартом семейства алгоритмов SHA, разработанным (NIST). Он обеспечивает хороший уровень безопасности.

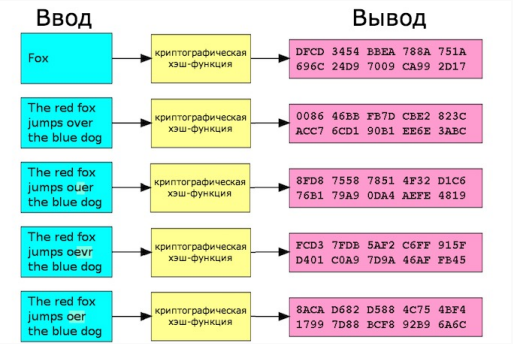
BLAKE2: Это высокопроизводительный хеш-алгоритм, который предлагает хороший баланс между безопасностью и производительностью. BLAKE2 обеспечивает высокий уровень стойкости к различным видам атак, включая коллизии.

Слабые алгоритмы:

MD5 : MD5 был широко использован в прошлом, но он считается слабым, так как были найдены коллизии.

SHA-1 : SHA-1 также считается слабым и устаревшим. Были обнаружены коллизии, после чего NIST рекомендует избегать использования SHA-1 в криптографических приложениях.

CRC : CRC легко поддается коллизиям и не предназначен для криптографических задач.



Teach
Me
Skills

Спасибо за внимание!