

< Teach
Me
Skills />

Кодинг и уязвимости

Вопросы по предыдущим темам или ДЗ

Mini-quiz по прошлым темам:

1. В каком формате передаются данные в REST API?
2. Какие 2 основных метода используются в REST API?
3. В чем заключается основное назначение расследования инцидента?
4. Какие системы вы бы подключили в SOAR? И какие playbook реагирования создавали?

Mini-quiz по новой теме:

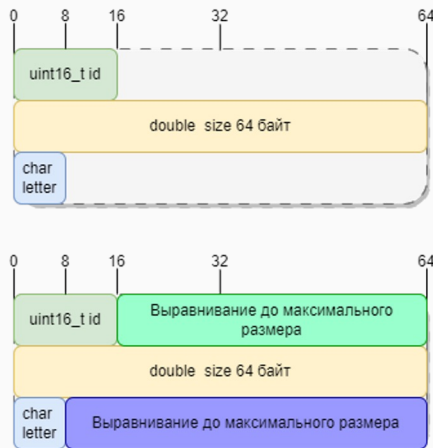
1. В чем могут заключаться проблемы при использовании интерпретируемых ЯП?
2. К чему может привести отсутствие обработчиков ошибок?
3. В чем заключается основная проблема при отсутствии валидации введенных данных?
4. В чем отличие между переполнением буфера и инъекцией в код?

План занятия

1. Рассмотрим возможные варианты атак на приложения
2. Рассмотрим на примере атаки на приложения, которые возможно произвести из-за ошибок в программировании
3. Рассмотрим особенности низкоуровневых ЯП
4. Рассмотрим алгоритмы хэширования
5. Так же изучим остальные виды ошибок при программировании
6. Разберемся с процессом внедрения шеллкодов и библиотек

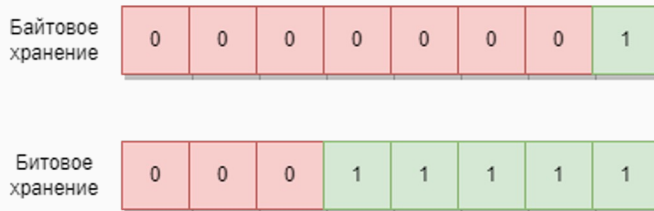
1. Выравнивание

```
struct example{  
    uint16_t id = 23;  
    double size = 1.1;  
    char letter = 'q';  
};
```



Выравнивание структуры — это процесс добавления дополнительных байтов в структуру данных, чтобы каждый элемент данных имел адрес, кратный размеру наибольшего элемента в структуре. Это делается для оптимизации доступа к памяти и улучшения производительности.

2. Хранение bool



Хранение типа данных **bool** в памяти зависит от компилятора. Наиболее распространенными типами хранения - являются:

байтовое (8 бит):

- Наименьшее количество памяти, которое может быть выделено для хранения **bool**, это 1 байт. Однако, по факту, для экономии памяти часто используется только 1 бит, а оставшиеся 7 бит могут оставаться неиспользуемыми.

битовое:

- Иногда, чтобы сэкономить память, особенно при хранении большого количества **bool** значений, программисты могут использовать битовые поля, битовые маски или другие методы для упаковки нескольких **bool** значений в один байт.

Особенности программирования #2

3. Проецирование процесса в память

1. Загрузка PE-файла:

ОС загружает PE-файл в память.

2. Инициализация процесса:

ОС создает процесс для исполнения приложения.

3. Создание виртуального адресного пространства:

Для процесса выделяется виртуальное пространство, в нем будет исполняться приложение.

4. Размещение загруженного PE-файла в памяти:

ОС размещает загруженный PE-файл в виртуальном адресном пространстве процесса, включая секции `.text`, `.data`, `.bss`, и др.

5. Инициализация структур данных:

Инициализируются структуры данных, такие как импорт (`.idata`), экспорт, ресурсы и т.д.

6. Инициализация стека и регистров:

Инициализируется стек и регистры процессора, подготавливаясь к выполнению кода.

7. Выполнение точки входа:

Выполняется код, указанный в поле "AddressOfEntryPoint" заголовка PE-файла. Обычно функция `main` для программ на языке C или C++.

8. Завершение программы:

По завершении выполнения кода приложения происходит завершение процесса. Это может быть освобождение ресурсов, закрытие файлов, высвобождение памяти и так далее.

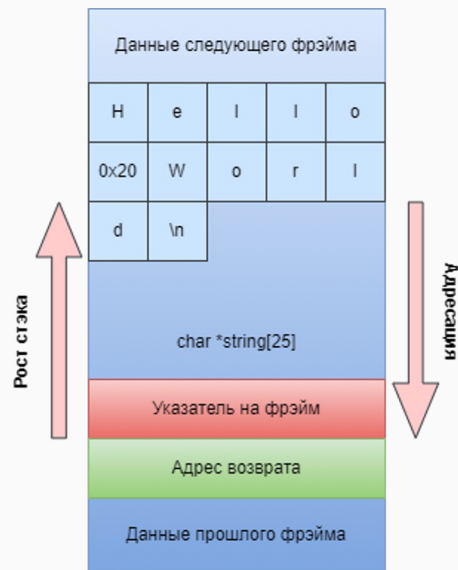
Имя	Назначение	Адреса
-----	Заголовки приложения и ссылки на таблицы импорта/экспорта	0x400000 0x401000 0x001000
.text	Секция кода приложения, самисписный + сгенерированный	0x401000 0x404000 0x003000
.data	Инициализированные данные	0x404000 0x405000 0x001000
.rdata	Данные только для чтения, в основном данные компилятора	0x405000 0x406000 0x001000
.bss	Данные, которые не нуждаются в явной инициализации	0x406000 0x407000 0x001000
.idata	Содержит информацию, о библиотеках	0x407000 0x408000 0x001000
.CRT	Код и данные, связанные с инициализацией рантайма	0x408000 0x409000 0x001000

Адрес	Шестнадцатеричное	ASCII
00400000	4D EA 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZ.....yy..
00400010	B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00@.....
00400020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00400030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00400040	0E 1F 8A 0E 00 B4 09 CD 21 B8 01 4C 21 54 68I.Liith
00400050	69 73 20 70 72 67 72 61 69 20 63 6E 6E 6F is	program canno
00400060	74 20 62 6E 20 72 75 6E 20 69 6E 20 44 4F 53 20	t be run in DOS
00400070	6D 6F 64 65 2E 00 00 0A 24 00 00 00 00 00 00 00	mode...\$.....

00401000	75	push esp
00401001	8B5E	mov esp,esp
00401002	8B5E	mov esp,esp
00401003	8B5E	mov esp,esp
00401004	8B5E	mov esp,esp
00401005	8B5E	mov esp,esp
00401006	8B5E	mov esp,esp
00401007	8B5E	mov esp,esp
00401008	8B5E	mov esp,esp
00401009	8B5E	mov esp,esp
0040100A	8B5E	mov esp,esp
0040100B	8B5E	mov esp,esp
0040100C	8B5E	mov esp,esp
0040100D	8B5E	mov esp,esp
0040100E	8B5E	mov esp,esp
0040100F	8B5E	mov esp,esp
00401010	8B5E	mov esp,esp
00401011	8B5E	mov esp,esp
00401012	8B5E	mov esp,esp
00401013	8B5E	mov esp,esp
00401014	8B5E	mov esp,esp
00401015	8B5E	mov esp,esp
00401016	8B5E	mov esp,esp
00401017	8B5E	mov esp,esp
00401018	8B5E	mov esp,esp
00401019	8B5E	mov esp,esp
0040101A	8B5E	mov esp,esp
0040101B	8B5E	mov esp,esp
0040101C	8B5E	mov esp,esp
0040101D	8B5E	mov esp,esp
0040101E	8B5E	mov esp,esp
0040101F	8B5E	mov esp,esp
00401020	8B5E	mov esp,esp
00401021	8B5E	mov esp,esp
00401022	8B5E	mov esp,esp
00401023	8B5E	mov esp,esp
00401024	8B5E	mov esp,esp
00401025	8B5E	mov esp,esp
00401026	8B5E	mov esp,esp
00401027	8B5E	mov esp,esp
00401028	8B5E	mov esp,esp
00401029	8B5E	mov esp,esp
0040102A	8B5E	mov esp,esp
0040102B	8B5E	mov esp,esp
0040102C	8B5E	mov esp,esp
0040102D	8B5E	mov esp,esp
0040102E	8B5E	mov esp,esp
0040102F	8B5E	mov esp,esp
00401030	8B5E	mov esp,esp
00401031	8B5E	mov esp,esp
00401032	8B5E	mov esp,esp
00401033	8B5E	mov esp,esp
00401034	8B5E	mov esp,esp
00401035	8B5E	mov esp,esp
00401036	8B5E	mov esp,esp
00401037	8B5E	mov esp,esp
00401038	8B5E	mov esp,esp
00401039	8B5E	mov esp,esp
0040103A	8B5E	mov esp,esp
0040103B	8B5E	mov esp,esp
0040103C	8B5E	mov esp,esp
0040103D	8B5E	mov esp,esp
0040103E	8B5E	mov esp,esp
0040103F	8B5E	mov esp,esp
00401040	8B5E	mov esp,esp
00401041	8B5E	mov esp,esp
00401042	8B5E	mov esp,esp
00401043	8B5E	mov esp,esp
00401044	8B5E	mov esp,esp
00401045	8B5E	mov esp,esp
00401046	8B5E	mov esp,esp
00401047	8B5E	mov esp,esp
00401048	8B5E	mov esp,esp
00401049	8B5E	mov esp,esp
0040104A	8B5E	mov esp,esp
0040104B	8B5E	mov esp,esp
0040104C	8B5E	mov esp,esp
0040104D	8B5E	mov esp,esp
0040104E	8B5E	mov esp,esp
0040104F	8B5E	mov esp,esp
00401050	8B5E	mov esp,esp
00401051	8B5E	mov esp,esp
00401052	8B5E	mov esp,esp
00401053	8B5E	mov esp,esp
00401054	8B5E	mov esp,esp
00401055	8B5E	mov esp,esp
00401056	8B5E	mov esp,esp
00401057	8B5E	mov esp,esp
00401058	8B5E	mov esp,esp
00401059	8B5E	mov esp,esp
0040105A	8B5E	mov esp,esp
0040105B	8B5E	mov esp,esp
0040105C	8B5E	mov esp,esp
0040105D	8B5E	mov esp,esp
0040105E	8B5E	mov esp,esp
0040105F	8B5E	mov esp,esp
00401060	8B5E	mov esp,esp
00401061	8B5E	mov esp,esp
00401062	8B5E	mov esp,esp
00401063	8B5E	mov esp,esp
00401064	8B5E	mov esp,esp
00401065	8B5E	mov esp,esp
00401066	8B5E	mov esp,esp
00401067	8B5E	mov esp,esp
00401068	8B5E	mov esp,esp
00401069	8B5E	mov esp,esp
0040106A	8B5E	mov esp,esp
0040106B	8B5E	mov esp,esp
0040106C	8B5E	mov esp,esp
0040106D	8B5E	mov esp,esp
0040106E	8B5E	mov esp,esp
0040106F	8B5E	mov esp,esp
00401070	8B5E	mov esp,esp
00401071	8B5E	mov esp,esp
00401072	8B5E	mov esp,esp
00401073	8B5E	mov esp,esp
00401074	8B5E	mov esp,esp
00401075	8B5E	mov esp,esp
00401076	8B5E	mov esp,esp
00401077	8B5E	mov esp,esp
00401078	8B5E	mov esp,esp
00401079	8B5E	mov esp,esp
0040107A	8B5E	mov esp,esp
0040107B	8B5E	mov esp,esp
0040107C	8B5E	mov esp,esp
0040107D	8B5E	mov esp,esp
0040107E	8B5E	mov esp,esp
0040107F	8B5E	mov esp,esp
00401080	8B5E	mov esp,esp
00401081	8B5E	mov esp,esp
00401082	8B5E	mov esp,esp
00401083	8B5E	mov esp,esp
00401084	8B5E	mov esp,esp
00401085	8B5E	mov esp,esp
00401086	8B5E	mov esp,esp
00401087	8B5E	mov esp,esp
00401088	8B5E	mov esp,esp
00401089	8B5E	mov esp,esp
0040108A	8B5E	mov esp,esp
0040108B	8B5E	mov esp,esp
0040108C	8B5E	mov esp,esp
0040108D	8B5E	mov esp,esp
0040108E	8B5E	mov esp,esp
0040108F	8B5E	mov esp,esp
00401090	8B5E	mov esp,esp
00401091	8B5E	mov esp,esp
00401092	8B5E	mov esp,esp
00401093	8B5E	mov esp,esp
00401094	8B5E	mov esp,esp
00401095	8B5E	mov esp,esp
00401096	8B5E	mov esp,esp
00401097	8B5E	mov esp,esp
00401098	8B5E	mov esp,esp
00401099	8B5E	mov esp,esp
0040109A	8B5E	mov esp,esp
0040109B	8B5E	mov esp,esp
0040109C	8B5E	mov esp,esp
0040109D	8B5E	mov esp,esp
0040109E	8B5E	mov esp,esp
0040109F	8B5E	mov esp,esp
004010A0	8B5E	mov esp,esp
004010A1	8B5E	mov esp,esp
004010A2	8B5E	mov esp,esp
004010A3	8B5E	mov esp,esp
004010A4	8B5E	mov esp,esp
004010A5	8B5E	mov esp,esp
004010A6	8B5E	mov esp,esp
004010A7	8B5E	mov esp,esp
004010A8	8B5E	mov esp,esp
004010A9	8B5E	mov esp,esp
004010AA	8B5E	mov esp,esp
004010AB	8B5E	mov esp,esp
004010AC	8B5E	mov esp,esp
004010AD	8B5E	mov esp,esp
004010AE	8B5E	mov esp,esp
004010AF	8B5E	mov esp,esp
004010B0	8B5E	mov esp,esp
004010B1	8B5E	mov esp,esp
004010B2	8B5E	mov esp,esp
004010B3	8B5E	mov esp,esp
004010B4	8B5E	mov esp,esp
004010B5	8B5E	mov esp,esp
004010B6	8B5E	mov esp,esp
004010B7	8B5E	mov esp,esp
004010B8	8B5E	mov esp,esp
004010B9	8B5E	mov esp,esp
004010BA	8B5E	mov esp,esp
004010BB	8B5E	mov esp,esp
004010BC	8B5E	mov esp,esp
004010BD	8B5E	mov esp,esp
004010BE	8B5E	mov esp,esp
004010BF	8B5E	mov esp,esp
004010C0	8B5E	mov esp,esp
004010C1	8B5E	mov esp,esp
004010C2	8B5E	mov esp,esp
004010C3	8B5E	mov esp,esp
004010C4	8B5E	mov esp,esp
004010C5	8B5E	mov esp,esp
004010C6	8B5E	mov esp,esp
004010C7	8B5E	mov esp,esp
004010C8	8B5E	mov esp,esp
004010C9	8B5E	mov esp,esp
004010CA	8B5E	mov esp,esp
004010CB	8B5E	mov esp,esp
004010CC	8B5E	mov esp,esp
004010CD	8B5E	mov esp,esp
004010CE	8B5E	mov esp,esp
004010CF	8B5E	mov esp,esp
004010D0	8B5E	mov esp,esp
004010D1	8B5E	mov esp,esp
004010D2	8B5E	mov esp,esp
004010D3	8B5E	mov esp,esp
004010D4	8B5E	mov esp,esp
004010D5	8B5E	mov esp,esp
004010D6	8B5E	mov esp,esp
004010D7	8B5E	mov esp,esp
004010D8	8B5E	mov esp,esp
004010D9	8B5E	mov esp,esp
004010DA	8B5E	mov esp,esp
004010DB	8B5E	mov esp,esp
004010DC	8B5E	mov esp,esp
004010DD	8B5E	mov esp,esp
004010DE	8B5E	mov esp,esp
004010DF	8B5E	mov esp,esp
004010E0	8B5E	mov esp,esp
004010E1	8B5E	mov esp,esp
004010E2	8B5E	mov esp,esp
004010E3	8B5E	mov esp,esp
004010E4	8B5E	mov esp,esp
004010E5	8B5E	mov esp,esp
004010E6	8B5E	mov esp,esp
004010E7	8B5E	mov esp,esp
004010E8	8B5E	mov esp,esp
004010E9	8B5E	mov esp,esp
004010EA	8B5E	mov esp,esp
004010EB	8B5E	mov esp,esp
004010EC	8B5E	mov esp,esp
004010ED	8B5E	mov esp,esp
004010EE	8B5E	mov esp,esp
004010EF	8B5E	mov esp,esp
004010F0	8B5E	mov esp,esp
004010F1	8B5E	mov esp,esp
004010F2	8B5E	mov esp,esp
004010F3	8B5E	mov esp,esp
004010F4	8B5E	mov esp,esp
004010F5	8B5E	mov esp,esp
004010F6	8B5E	mov esp,esp
004010F7	8B5E	mov esp,esp
004010F8	8B5E	mov esp,esp
004010F9	8B5E	mov esp,esp
004010FA	8B5E	mov esp,esp
004010FB	8B5E	mov esp,esp
004010FC	8B5E	mov esp,esp
004010FD	8B5E	mov esp,esp
004010FE	8B5E	mov esp,esp
004010FF	8B5E	mov esp,esp
00401100	8B5E	mov esp,esp
00401101	8B5E	mov esp,esp
00401102	8B5E	mov esp,esp
00401103	8B5E	mov esp,esp
00401104	8B5E	mov esp,esp
00401105	8B5E	mov esp,esp
00401106	8B5E	mov esp,esp
00401107	8B5E	mov esp,esp
00401108	8B5E	mov esp,esp
00401109	8B5E	mov esp,esp
0040110A	8B5E	mov esp,esp
0040110B	8B5E	mov esp,esp
0040110C	8B5E	mov esp,esp
0040110D	8B5E	mov esp,esp
0040110E	8B5E	mov esp,esp
0040110F	8B5E	mov esp,esp
00401110	8B5E	mov esp,esp
00401111	8B5E	mov esp,esp
00401112	8B5E	mov esp,esp
00401113	8B5E	mov esp,esp
00401114	8B5E	mov esp,esp
00401115	8B5E	mov esp,esp
00401116	8B5E	mov esp,esp
00401117	8B5E	mov esp,esp
00401118	8B5E	mov esp,esp
00401119	8B5E	mov esp,esp
0040111A	8B5E	mov esp,esp
0040111B	8B5E	mov esp,esp
0040111C	8B5E	mov esp,esp
0040111D	8B5E	mov esp,esp
0040111E	8B5E	mov esp,esp
0040111F	8B5E	mov esp,esp
00401120	8B5E	mov esp,esp
00401121	8B5E	mov esp,esp
00401122	8B5E	mov esp,esp
00401123	8B5E	mov esp,esp
00401124	8B5E	mov esp,esp
00401125	8B5E	mov esp,esp
00401126	8B5E	mov esp,esp
00401127	8B5E	mov esp,

Как работает буфер

В контексте программирования и информационной безопасности, термин **буфер** (buffer) часто означает область памяти, предназначенную для временного хранения данных. Буферы используются в различных аспектах программирования и компьютерных систем. В программировании представляет собой массив или структуру данных, используемую для временного хранения информации



Буфер в c/c++

```
//Буфер типа символьной строки
char *buff1;
//Буфер типа массив символов размером 10
char buff2[10];
//Буфер типа массив символов созданный динамически и размером size
int buff3 = new char[size];
```

Буфер в Python

```
answer = requests.get(url)
// Динамический буфер в Python
buff = answer.get('last_analysis_results')
```

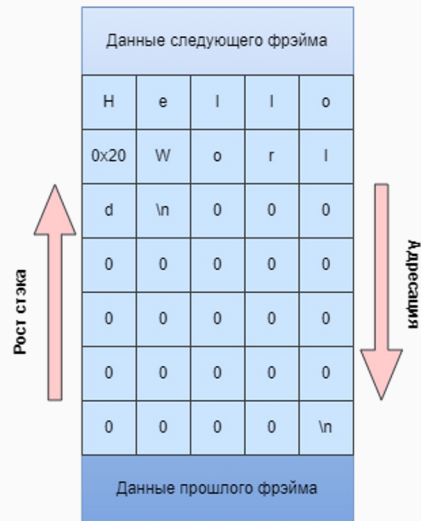

Переполнение буфера

Переполнение буфера (Buffer Overflow) — это тип атаки на программное обеспечение, при котором данные, превышающие размер выделенного буфера, переписывают другие области памяти.

Пример кода подверженного переполнению буфера:

```
#include <string.h>

int main(int argc, char *argv[]) {
    char buf[100];
    strcpy(buf, argv[1]);
    return 0;
}
```



База фрейма и адрес возврата – затерты!

1. Исполнение вредоносного кода:

Злоумышленники могут внедрить вредоносный код, который будет выполнен после успешного переполнения буфера.

2. Отказ в обслуживании:

Переполнение буфера может привести к аварийному завершению программы, вызывая отказ в обслуживании.

3. Утечка конфиденциальных данных:

Злоумышленники могут получить доступ к конфиденциальным данным, переписывая содержимое буфера.

Форматная строка

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

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

scanf("%d", &number);

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

```
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
```

Уязвимости форматной строки

Данная уязвимость связана с функциями форматного вывода и форматирования строки - `printf`, `fprintf`, `sprintf`, `snprintf`.

Все эти функции имеют неопределенное количество параметров, формата: “**const char *format**, ...”.

При этом число параметров и их интерпретация определяются посредством строки форматных спецификаторов *format*.

```
1  #include <stdio.h>
2  #include <iostream>
3
4  void vulnerable_function(char *user_input) {
5      printf(user_input);
6  }
7
8  int main() {
9      char input[] = "";
10     std::cin >> input; // Ввод данных строки формата %s%n
11     vulnerable_function(input); // Вывод введенных данных
12     return 0;
13 }
```

Так как **printf** воспринимает форматные спецификаторы, если мы передадим на ввод строку формата `%x%x%x`.

Printf - не знает о количестве и наличии параметров. И поэтому он выдает 3 параметра в 16-ном виде из стека.

Уязвимости при загрузке файлов

1. Недостаточная валидация файлов:

1. Проблема: Необходимо проверять типы и содержимое файлов, чтобы убедиться, что загружаемые файлы соответствуют ожидаемым форматам.
2. Решение: Используйте механизмы валидации, такие как проверка MIME-типа, анализ заголовков файлов, и другие методы, чтобы гарантировать правильность типа файла.

2. Вредоносные файлы с двусмысленными расширениями:

1. Проблема: Злоумышленники могут попытаться обмануть систему, изменяя расширение файла на неопасное, но используя вредоносное содержимое.
2. Решение: Проверяйте не только расширение файла, но и его содержимое для определения реального типа.

3. Подделка заголовков (Content-Disposition):

1. Проблема: Злоумышленники могут попытаться подделать заголовки, чтобы скрыть настоящий характер файла.
2. Решение: Проверяйте заголовки, такие как Content-Disposition, чтобы удостовериться, что они соответствуют ожидаемым значениям.

4. Переполнение буфера:

1. Проблема: Если веб-приложение не ограничивает размер загружаемых файлов, это может привести к атакам переполнения буфера.
2. Решение: Установите ограничения на размер файлов для предотвращения атак на переполнение буфера.

5. Исполнение вредоносного кода:

1. Проблема: Загруженные файлы могут содержать вредоносный код, который выполняется на сервере или клиенте.
2. Решение: Используйте механизмы антивирусной проверки и регулярные обновления для обнаружения и предотвращения вредоносного кода.

```
int (*ret)() = (int(*)())code;
```

```
ret();
```

Инъекция кода: теория

Инъекция DLL бывает нескольких видов:

1. Remote Thread Injection:

Описание: Злоумышленник создает удаленный поток в процессе целевого приложения и внедряет свой код в виде DLL в адресное пространство этого процесса.

Методы: *CreateRemoteThread*, *CreateThread*, *QueueUserAPC*, *NtCreateThreadEx*.

2. LoadLibrary Injection:

Описание: Злоумышленник использует функцию **LoadLibrary** для загрузки своей DLL в адресное пространство процесса. Этот метод требует изменения контекста целевого процесса.

Методы: *CreateRemoteThread*, *SetWindowsHookEx*, *AppInit_DLLs*.

3. Reflective DLL Injection:

- **Описание:** Используют технику, при которой DLL самовоспроизводится в памяти, без необходимости сохранения на диск. Во время выполнения DLL загружается и исполняется из памяти, что делает ее более скрытой.

Инструменты: такие как Metasploit Meterpreter, используют рефлексивные инъекции.



Инъекция кода

Инъекция команд в Python

Команда `os.system` позволяет выполнять в операционной системе, команды переданные в качестве текстовой строки:

```
import os
user_input = "foo && cat /etc/passwd" # value supplied by user
os.system("grep -R {} {}".format(user_input))
```

Команда `subprocess` позволяет создавать новые процессы, метод `call` выполняет переданную текстовую строку:

```
subprocess.call('echo $HOME', shell=True)
```


Инъекция кода

Как лечить

Общие правила:

- Никогда не доверяйте данным, передаваемым пользователями!
- Везде, где это возможно, не разрешайте выполнение произвольных команд. Вместо этого используйте списки*
- Используйте рекомендованные сообществом безопасные команды и параметры
- Используйте специальные библиотеки для контроля пользовательского ввода

Особенности работы с процессами

- избегать использования дочерних процессов
- ограничить привилегии дочернего процесса
- запуск дочернего процесса в изолированной среде

```
const Sandbox = require("sandbox")
, s = new Sandbox()

s.run( "process.platform", function( output ) {
  console.log(output);
  //output=NULL
})
```

Десериализация

Десериализация – это процесс декодирования данных JSON в объекты Python. Модуль json предоставляет два метода load() и loads(), которые используются для преобразования данных JSON в фактическую объектную форму Python.

Модуль pickle:

- отсутствие контроля за целостностью данных / объектов
- отсутствие контроля над размером данных или системных ограничений
- код оценивается без мер безопасности
- строки кодируются / декодируются без проверки

```
import pickle

object_list=['learning','python','security']
with open('data.pickle', 'wb') as file:
    pickle.dump(object_list, file)
```

Десериализация

Как лечить

Общие правила:

- Никогда не доверяйте данным, передаваемым пользователями!
- Никогда не производите десериализацию данных из ненадёжного источника с помощью pickle.
- Используйте другой шаблон сериализации, например, JSON.

Shellcode

Shell-код - набор машинных команд, позволяющий получить доступ к командному интерпретатору bash/cmd

Шелл-код обычно внедряется в память эксплуатируемой программы, после чего на него передается управление путём переполнения стека, или при переполнении буфера в куче, или используя атаки форматной строки.

Пример shellcode

```
00000000  33C9          xor ecx,ecx
00000002  648B4130      mov eax,[fs:ecx+0x30]
00000006  8B400C        mov eax,[eax+0xc]
00000009  8B7014        mov esi,[eax+0x14]
0000000C  AD           lodsd
0000000D  96           xchg eax,esi
0000000E  AD           lodsd
0000000F  8B5810        mov ebx,[eax+0x10]
00000012  8B533C        mov edx,[ebx+0x3c]
00000015  03D3          add edx,ebx
00000017  8B5278        mov edx,[edx+0x78]
0000001A  03D3          add edx,ebx
0000001C  8B7220        mov esi,[edx+0x20]
0000001F  03F3          add esi,ebx
00000021  33C9          xor ecx,ecx
```

***В более широком смысле** shell-код — это любой код, который используется как **payload** и представляет собой последовательность машинных команд, которую выполняет уязвимое приложение

Небезопасный запуск скриптов

При запуске файла без абсолютного пути может использоваться небезопасное значение переменной PATH или запуск файла из другого каталога.

При наличии права на запись у атакующего возможно выполнение произвольного кода.

```
~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/local/games
:/usr/games
```

Запуск скриптов и установка обновлений из не доверенных каталогов.

```
~$ cd Downloads
~/Downloads$ python -m pip install ./downloaded-package.whl
```


Небезопасный запуск скриптов

Лечение

- каждая запись в `sys.path` является безопасным местом
- каталог, в котором находится главный скрипт, всегда находится в `sys.path`
- при прямом вызове `python` текущий каталог рассматривается как расположение основного скрипта
- вводить путь к сценарию, а не к модулю
- избегать использования недоверенных каталогов в качестве текущего

Атаки на XML

При парсинге XML в Python обычно используют стандартную библиотеку XML. Однако, стандартная библиотека уязвима к некоторым DoS атакам.

Атака Billion Laughs

```
<?xml version="1.0"?>
<!DOCTYPE lolz [
<!ENTITY lol "lol">
<!ENTITY lol2 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">
<!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
<!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
<!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">
<!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">
<!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;">
<!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">
]>
<lolz>&lol8;</lolz>
```

Загрязненный site-packages, устаревшие пакеты

- установка сторонних пакетов в ваш site-packages, будь то в виртуальном окружении или глобальный site-packages, обеспечивает вам дыры в безопасности в этих пакетах
- уязвимости в сторонних пакетах, которые в свою очередь могут обеспечить дыры
- зависимости от зависимостей, которые тоже могут иметь дыры
- устаревшие пакеты

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.

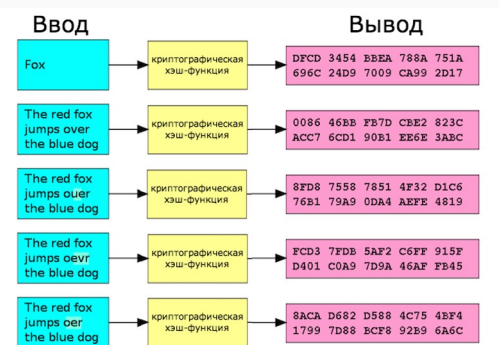
BLAKE2: Это высокопроизводительный хеш-алгоритм, который предлагает хороший баланс между безопасностью и производительностью. BLAKE2 обеспечивает высокий уровень стойкости к различным видам атак, включая коллизии.

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.



Проблемы алгоритмов хэширования

SSDEEP – функция не четкого хэширования которая - создает размытый хеш, который устойчив к некоторым видам изменений в данных. Это особенно полезно при сравнении файлов, которые могли быть незначительно изменены.

Сравнение хешей: Сравнение SSDEEP-хешей может дать представление о том, насколько два файла похожи друг на друга. Чем больше общих блоков данных у файлов, тем выше степень схожести по хешам.

Использование в целях обнаружения изменений и малых различий: SSDEEP часто применяется для обнаружения малых изменений в файлах, что может быть полезно в областях, таких как ИБ и форензика.



SSDEEP - Fuzzing Hashing Techniques
to Detect Unknown Malware

Улучшение механизма аутентификации

- добавление даты создания билета FA
- добавление количества перевыпусков у билета FA
- добавление уникального идентификатора сеанса в билет FA
- непредсказуемый токен CSRF при каждом выпуске билета
- более частый запуск проверок учетных данных
- добавление мастер-ключа на основе пользовательских данных

Механизм сброса пароля

- решение пользователем капчи
- создание и отправка токена пользователю
- валидация токена после его введения пользователем
- задание нового пароля
- детальный аудит

Почему программы на С/С++ особенно уязвимы

- Имеется прямой доступ к памяти
- Нет дополнительного уровня безопасности в виде виртуальной машины
- Могут использоваться устаревшие (небезопасные) функции
- При исполнении кода нет контроля выхода за границу памяти

Спасибо за внимание!