C++ Russia 2021

Ошибки C++, приводящие к уязвимостям, и их митигация на KasperskyOS

Сергей Талантов
Архитектор ПО
Чемпион безопасности

KasperskyOS

Содержание

Основные определения

Убедимся, что говорим на одном языке

Баг (bug)
ошибка в программе, при которой она выдает неожиданное поведение и результат

Баг (bug)
ошибка в программе, при которой она выдает неожиданное поведение и результат

Уязвимость (vulnerability)
недостаток в системе, используя который, можно намеренно нарушить её целостность и вызвать неправильную работу

Баг (bug)
ошибка в программе, при которой она выдает неожиданное поведение и результат

Уязвимость (vulnerability)
недостаток в системе, используя который, можно намеренно нарушить её целостность и вызвать неправильную работу

Не каждый баг приводит к уязвимости, не каждая уязвимость вызвана багом.

Не каждая уязвимость приводит к эксплойту.

Эксплойт (exploit)
вредоносный код или приложение, который использует уязвимость в ПО

Не каждая уязвимость приводит к эксплойту.

Чем больше поверхность атаки, тем выше вероятность эксплойта.

Эксплойт (exploit)
вредоносный код или приложение, который использует уязвимость в ПО

Поверхность атаки (attack surface)
общее количество возможных уязвимых мест

Риск безопасности (security risk)

вероятные потери в результате инцидентов

Риск безопасности (security risk)
вероятные потери в результате инцидентов

Митигация рисков безопасности (security mitigation)
добавляемый в продукт механизм защиты от эксплойтов

Риск безопасности (security risk)
вероятные потери в результате инцидентов

Митигация рисков безопасности (security mitigation)
добавляемый в продукт механизм защиты от эксплойтов

Риски безопасности можно и нужно митигировать.

Кто такие чемпионы безопасности

Рассмотрим роль чемпиона безопасности в контексте процесса безопасной разработки ПО

Security Development Lifecycle (SDL) [8] - процесс обеспечения безопасности в сфере разработки программного обеспечения.

Разработан в **Microsoft** используется в **Лаборатории Касперского**.

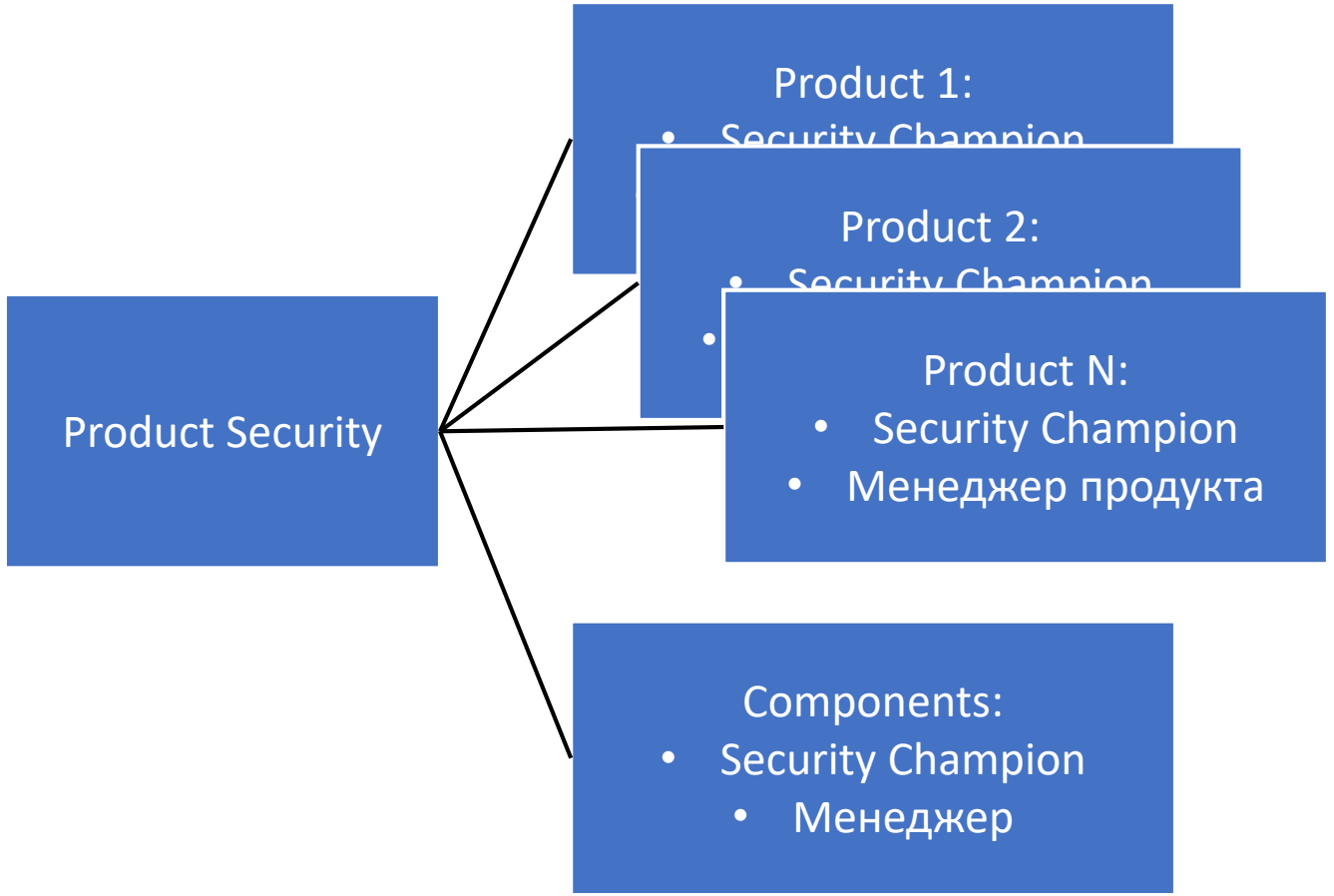
Процесс SDL призван уменьшить число уязвимостей в программном обеспечении и уровень их серьезности. SDL позволяет обеспечить безопасность и конфиденциальность **на всех этапах процесса разработки.**

Кто такие чемпионы безопасности

SDL представляет собой набор обязательных действий по обеспечению безопасности, перечисленных в порядке их выполнения и сгруппированных по этапам стандартного жизненного цикла разработки ПО.

Люди и роли:

1. Чемпион безопасности (security champion)
2. Менеджер продукта
3. Отдел безопасности продуктов



Чемпион безопасности – член команды разработки, имеющий экспертизу как разработчика, так и инженера по безопасности [9].

Координацию и отслеживание вопросов безопасности для проекта.

Раннее выявление и решение проблем безопасности.

Эскалация проблем менеджеру проекта или отделу безопасности.

Проблемный код и его последствия

Разбираем на примерах строки,
контейнеры, память, исключения...

Пример проблемного кода №1


```cpp
#include <cstdlib>
#include <string>

void f() {
  std::string tmp(std::getenv("TMP"));
  if (!tmp.empty()) {
    // ...
  }
}
```

Инициализация строки нулевым указателем

```cpp
#include <cstdlib>
#include <string>

void f() {
  std::string tmp(std::getenv("TMP"));
  if (!tmp.empty()) {
    // ...
  }
}
```

Риски:

1. UB
2. Падение
3. Потенциально запуск чужого кода, если по нулевому адресу замаплена память [10]

Проверяем указатель

```cpp
#include <cstdlib>
#include <string>

void f() {
  const char *tmpPtrVal = std::getenv("TMP");
  std::string tmp(tmpPtrVal ? tmpPtrVal : "");
  if (!tmp.empty()) {
    // ...
  }
}
```

Пример проблемного кода №2


```cpp
#include <string>

void f(std::string& s) {
  for (auto it = std::begin(s); it != std::end(s); ++it) {
    if (*it == ' ')
      s.erase(it);
  }
}
```

Инвалидация итераторов

```cpp
#include <string>

void f(std::string& s) {
  for (auto it = std::begin(s); it != std::end(s); ++it) {
    if (*it == ' ')
      s.erase(it);
  }
}
```

Риски:
Запуск чужого кода [1]

Следим за итераторами или используем стандартные алгоритмы

```cpp
#include <string>
#include <algorithm>

void f(std::string& s) {
  for (auto it = std::begin(s); it != std::end(s);) {
    if (*it == ' ')
      it = s.erase(it);
    else
      ++it;
  }
}

void f_better(std::string& s) {
  s.erase(std::remove(std::begin(s), std::end(s), ' '),
    std::end(s));
}
```

Пример проблемного кода №3


```
1 #include <string>
2 #include <locale>
3
4 void capitalize(std::string &s) {
5   std::locale loc;
6   s.front() = std::use_facet<std::ctype<char>>(loc).
7     toupper(s.front());
8 }
```

Выход за границы

```
1 #include <string>
2 #include <locale>
3
4 void capitalize(std::string &s) {
5   std::locale loc;
6   s.front() = std::use_facet<std::ctype<char>>(loc).
7     toupper(s.front());
8 }
```

Риски:

1. UB
2. Падение
3. Запуск чужого кода

Проверяем, что границы допустимы

```
1 #include <string>
2 #include <locale>
3
4 void capitalize(std::string &s) {
5   if (s.empty()) {
6     return;
7   }
8
9   std::locale loc;
10  s.front() = std::use_facet<std::ctype<char>>(loc).
11    toupper(s.front());
12 }
```

Пример проблемного кода №4


```cpp
#include <algorithm>
#include <vector>

void f(const std::vector<int> &src) {
  std::vector<int> dest;
  std::copy(src.begin(), src.end(), dest.begin());
  // ...
}
```

Пример проблемного кода №4

Невалидные буферы

```
1 #include <algorithm>
2 #include <vector>
3
4 void f(const std::vector<int> &src) {
5   std::vector<int> dest;
6   std::copy(src.begin(), src.end(), dest.begin());
7   // ...
8 }
```

Риски:

1. Падение
2. Запуск чужого кода

Используем валидные буферы

```
1 #include <algorithm>
2 #include <vector>
3
4 void f(const std::vector<int> &src) {
5   std::vector<int> dest(src.size());
6   std::copy(src.begin(), src.end(), dest.begin());
7   // ...
8 }
9
10 void f_reserve(const std::vector<int> &src) {
11   std::vector<int> dest;
12   dest.reserve(src.size());
13   std::copy(src.begin(), src.end(), std::back_inserter(dest));
14   // ...
15 }
16
17 void f_assign(const std::vector<int> &src) {
18   std::vector<int> dest(src);
19   // ...
20 }
```

Пример проблемного кода №5


```
 1 class Base {
 2 public:
 3   void DoIt() {/*...*/};
 4 };
 5
 6 class Derived : public Base {/*...*/};
 7
 8 void f(Base* elements, size_t size) {
 9   for (size_t i = 0; i < size; ++i)
10     elements[i].DoIt();
11 }
12
13 int main()
14 {
15   constexpr size_t size{ 10 };
16   Derived d[size];
17   f(d, size);
18   return 0;
19 }
```

Пример проблемного кода №5

Арифметика с указателями на полиморфные типы

```
1  class Base {
2  public:
3    void DoIt() {/*...*/};
4  };
5
6  class Derived : public Base {/*...*/};
7
8  void f(Base* elements, size_t size) {
9    for (size_t i = 0; i < size; ++i)
10     elements[i].DoIt();
11 }
12
13 int main()
14 {
15   constexpr size_t size{ 10 };
16   Derived d[size];
17   f(d, size);
18   return 0;
19 }
```

Риски:

1. UB
2. Падение
3. Повреждение данных
4. Запуск чужого кода

Используем стандартные контейнеры и умные указатели

```
1  void f(const std::vector<std::unique_ptr<Base>>& elements) {
2    for (auto& element: elements)
3      element->DoIt();
4  }
5
6  int main()
7  {
8    std::vector<std::unique_ptr<Base>> elements;
9    elements.push_back(std::make_unique<Derived>());
10   f(elements);
11   return 0;
12 }
```

Пример проблемного кода №6


```
 1 #include <string>
 2
 3 int f() {/*...*/}
 4
 5 const std::string global("...");
 6 const int i = f();
 7
 8 int main()
 9 try {
10   // ...
11 } catch(...) {
12 }
```

Пример проблемного кода №6

Исключения при инициализации глобальных объектов не перехватываются

```
 1 #include <string>
 2
 3 int f() {/*...*/}
 4
 5 const std::string global("...");
 6 const int i = f();
 7
 8 int main()
 9 try {
10   // ...
11 } catch(...) {
12 }
```

Риски:

1. Падение
2. Триггер для запуска чужого кода [13]

Не кидаем исключения при инициализации глобальных объектов

```
 1 #include <string>
 2
 3 int f() noexcept {/*...*/}
 4
 5 constexpr char global[] = "...";
 6 const int i = f();
 7
 8 int main()
 9 try {
10   // ...
11 } catch(...) {
12 }
```

Пример проблемного кода №7


```cpp
#include <cstdlib>
#include <string>

void f() {
  std::string id("ID"); // Holds the ID,
                        // starting with the characters "ID" followed
                        // by a random integer in the range [0-10000).
  id += std::to_string(std::rand() % 10000);
  // ...
}
```

Пример проблемного кода №7

Использование std::rand, который не гарантирует качество случайной последовательности

```cpp
#include <cstdlib>
#include <string>

void f() {
  std::string id("ID"); // Holds the ID,
                        // starting with the characters "ID" followed
                        // by a random integer in the range [0-10000).
  id += std::to_string(std::rand() % 10000);
  // ...
}
```

Риски:
Предсказуемые идентификаторы, пароли, секреты и т.д.

Используем функцию распределения и генератор случайных чисел с правильным исходным значением

```cpp
#include <random>
#include <string>

void f() {
  std::string id("ID"); // Holds the ID,
                        // starting with the characters "ID" followed
                        // by a random integer in the range [0-10000].
  std::uniform_int_distribution<int> distribution(0, 10000);
  std::random_device rd;
  std::mt19937 engine(rd());
  id += std::to_string(distribution(engine));
  // ...
}
```

Угрозы и их митигация

К чему приводят уязвимости и как их митигировать


Уязвимости, связанные с ошибками написания кода
Остальные уязвимости

Уязвимости приводят к угрозам. Классификация угроз по STRIDE [14]

S
T
R
I
D
E

Уязвимости приводят к угрозам. Классификация угроз по STRIDE [14]

	Угроза	Описание
S	Spoofing identify (спуфинг)	Незаконный доступ к данным пользователя (включая имя пользователя и пароль), используемыми для аутентификации, и их применения.
T		
R		
I		
D		
E		

Уязвимости приводят к угрозам. Классификация угроз по STRIDE [14]

	Угроза	Описание
S	Spoofing identify (спуфинг)	Незаконный доступ к данным пользователя (включая имя пользователя и пароль), используемыми для аутентификации, и их применения.
T	Tampering with data (незаконное изменение)	Вредоносное изменение данных.
R		
I		
D		
E		

Уязвимости приводят к угрозам. Классификация угроз по STRIDE [14]

	Угроза	Описание
S	Spoofing identify (спуфинг)	Незаконный доступ к данным пользователя (включая имя пользователя и пароль), используемыми для аутентификации, и их применения.
T	Tampering with data (незаконное изменение)	Вредоносное изменение данных.
R	Repudiation (отказ)	Отрицание пользователем, выполнившим какие-либо действия, факта их выполнения.
I		
D		
E		

Уязвимости приводят к угрозам. Классификация угроз по STRIDE [14]

	Угроза	Описание
S	Spoofing identify (спуфинг)	Незаконный доступ к данным пользователя (включая имя пользователя и пароль), используемыми для аутентификации, и их применения.
T	Tampering with data (незаконное изменение)	Вредоносное изменение данных.
R	Repudiation (отказ)	Отрицание пользователем, выполнившим какие-либо действия, факта их выполнения.
I	Information disclosure (раскрытие информации)	Раскрытие сведений пользователям, которые не должны иметь к ним доступ.
D		
E		

Уязвимости приводят к угрозам. Классификация угроз по STRIDE [14]

	Угроза	Описание
S	Spoofing identify (спуфинг)	Незаконный доступ к данным пользователя (включая имя пользователя и пароль), используемыми для аутентификации, и их применения.
T	Tampering with data (незаконное изменение)	Вредоносное изменение данных.
R	Repudiation (отказ)	Отрицание пользователем, выполнившим какие-либо действия, факта их выполнения.
I	Information disclosure (раскрытие информации)	Раскрытие сведений пользователям, которые не должны иметь к ним доступ.
D	Denial of service (отказ в обслуживании)	Отказ в обслуживании для допустимых пользователей
E		

Уязвимости приводят к угрозам. Классификация угроз по STRIDE [14]

	Угроза	Описание
S	Spoofing identify (спуфинг)	Незаконный доступ к данным пользователя (включая имя пользователя и пароль), используемыми для аутентификации, и их применения.
T	Tampering with data (незаконное изменение)	Вредоносное изменение данных.
R	Repudiation (отказ)	Отрицание пользователем, выполнившим какие-либо действия, факта их выполнения.
I	Information disclosure (раскрытие информации)	Раскрытие сведений пользователям, которые не должны иметь к ним доступ.
D	Denial of service (отказ в обслуживании)	Отказ в обслуживании для допустимых пользователей
E	Elevation of privilege (повышение привилегий)	Непривилегированный пользователь получает привилегированный доступ

Подход “Secure by design”

Архитектурная
митигация рисков

Подход “Secure by design”

Архитектурная митигация рисков

Основной фокус смещается с имплементации на проектирование

Подход “Secure by design”

Архитектурная митигация рисков

Основной фокус смещается с имплементации на проектирование

Следование специальным архитектурным паттернам делает систему условно безопасной [3, 4]

Подход “Secure by design”

Архитектурная митигация рисков

Основной фокус смещается с имплементации на проектирование

Следование специальным архитектурным паттернам делает систему условно безопасной [3, 4]

Подходящие архитектурные паттерны выбираются в момент проектирования и становятся основой во время разработки

Подход “Secure by design”

Архитектурная митигация рисков

Основной фокус смещается с имплементации на проектирование

Следование специальным архитектурным паттернам делает систему условно безопасной [3, 4]

Подходящие архитектурные паттерны выбираются в момент проектирования и становятся основой во время разработки

Применяется многоуровневая эшелонированная защита

KasperskyOS®

Вариант платформы для реализации безопасных решений с применением подхода “Secure by design” [12]

KasperskyOS®

Вариант платформы для реализации безопасных решений с применением подхода “Secure by design” [12]

Микроядро
Всего несколько десятков тысяч строк кода — поверхность атаки минимальна.

KasperskyOS®

Вариант платформы для реализации безопасных решений с применением подхода “Secure by design” [12]

Микроядро
Всего несколько десятков тысяч строк кода — поверхность атаки минимальна.

Строгая изоляция компонентов
Общение компонентов происходит только через единый механизм IPC.

KasperskyOS®

Вариант платформы для реализации безопасных решений с применением подхода “Secure by design” [12]

Микроядро
Всего несколько десятков тысяч строк кода — поверхность атаки минимальна.

Строгая изоляция компонентов
Общение компонентов происходит только через единый механизм IPC.

Монитор безопасности
контролирует каждый вызов IPC. Правила задаются политиками безопасности.

KasperskyOS®

Вариант платформы для реализации безопасных решений с применением подхода “Secure by design” [12]

Микроядро
Всего несколько десятков тысяч строк кода — поверхность атаки минимальна.

Строгая изоляция компонентов
Общение компонентов происходит только через единый механизм IPC.

Монитор безопасности
контролирует каждый вызов IPC. Правила задаются политиками безопасности.

Платформа для реализации кибериммунных решений

Кибериммунитет

Разбираем пример проектирования
кибериммунного решения

Нельзя просто так взять и запустить код на KasperskyOS, чтобы он стал кибериммунным

Продукт на KasperskyOS считается кибериммунным если:

Продукт на KasperskyOS считается кибериммунным если:

Есть описание целей и предположений безопасности

Цель безопасности – это функциональное требование, которые сохраняются во всех возможных сценариях работы системы, с учетом предположений безопасности.

Предположения безопасности – дополнительные ограничения или допущения, накладываемые на условия эксплуатации системы, которые влияют на достижение целей безопасности.

Продукт на KasperskyOS считается кибериммунным если:

Есть описание целей и предположений безопасности

Сделана изоляция доменов безопасности и введен независимый контроль всех взаимодействий

Цель безопасности – это функциональное требование, которые сохраняются во всех возможных сценариях работы системы, с учетом предположений безопасности.

Предположения безопасности – дополнительные ограничения или допущения, накладываемые на условия эксплуатации системы, которые влияют на достижение целей безопасности.

Продукт на KasperskyOS считается кибериммунным если:

Есть описание целей и предположений безопасности

Сделана изоляция доменов безопасности и введен независимый контроль всех взаимодействий

Обеспечено выполнение целей безопасности во всех возможных сценариях

Цель безопасности – это функциональное требование, которые сохраняются во всех возможных сценариях работы системы, с учетом предположений безопасности.

Предположения безопасности – дополнительные ограничения или допущения, накладываемые на условия эксплуатации системы, которые влияют на достижение целей безопасности.

Продукт на KasperskyOS считается кибериммунным если:

Есть описание целей и предположений безопасности

Сделана изоляция доменов безопасности и введен независимый контроль всех взаимодействий

Обеспечено выполнение целей безопасности во всех возможных сценариях

Обеспечена надежность доверенной вычислительной базы

Цель безопасности – это функциональное требование, которые сохраняются во всех возможных сценариях работы системы, с учетом предположений безопасности.

Предположения безопасности – дополнительные ограничения или допущения, накладываемые на условия эксплуатации системы, которые влияют на достижение целей безопасности.

Пример проектирования кибериммунного решения

Задача: реализовать подсистему нотификаций по протоколу MQTT

Пример проектирования кибериммунного решения

Задача: реализовать подсистему нотификаций по протоколу MQTT

MQTT (message queuing telemetry transport) —сетевой протокол, работающий поверх TCP/IP, ориентированный на обмен сообщениями между устройствами по принципу издатель-подписчик.

Пример проектирования кибериммунного решения

Задача: реализовать подсистему нотификаций по протоколу MQTT

MQTT (message queuing telemetry transport) —сетевой протокол, работающий поверх TCP/IP, ориентированный на обмен сообщениями между устройствами по принципу издатель-подписчик.

Определяем цели и предположения безопасности

Определяем цели и предположения безопасности

Цели безопасности:

1. Решение обеспечивает безопасный канал связи
2. Решение обеспечивает безопасное хранение внутренних данных (конфигурация, ключи, сертификаты)

Определяем цели и предположения безопасности

Цели безопасности:

1. Решение обеспечивает безопасный канал связи
2. Решение обеспечивает безопасное хранение внутренних данных (конфигурация, ключи, сертификаты)

Предположения безопасности:

1. Не рассматриваются аппаратные уязвимости
2. Не рассматривается физический доступ злоумышленника к аппаратной платформе

Выделяем доверенные и недоверенные компоненты

Выделяем доверенные и недоверенные компоненты

Доверенный компонент – часть кода, которой необходимо доверять для выполнения целей безопасности. Предъявляются особые требования по безопасности и качеству кода.

Выделяем доверенные и недоверенные компоненты

Доверенный компонент – часть кода, которой необходимо доверять для выполнения целей безопасности. Предъявляются особые требования по безопасности и качеству кода.

Недоверенный компонент – компонент не нарушающий цели безопасности даже при компрометации и дисфункции.

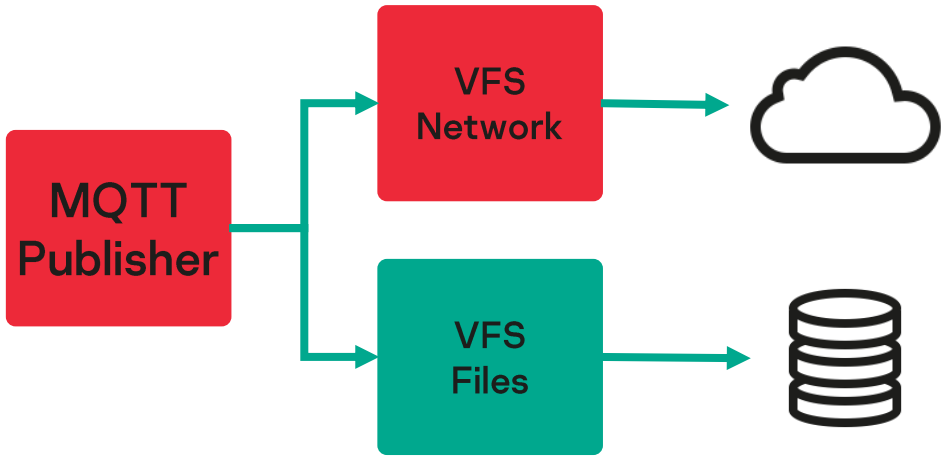
Выделяем доверенные и недоверенные компоненты

Доверенный компонент – часть кода, которой необходимо доверять для выполнения целей безопасности. Предъявляются особые требования по безопасности и качеству кода.

Недоверенный компонент – компонент не нарушающий цели безопасности даже при компрометации и дисфункции.

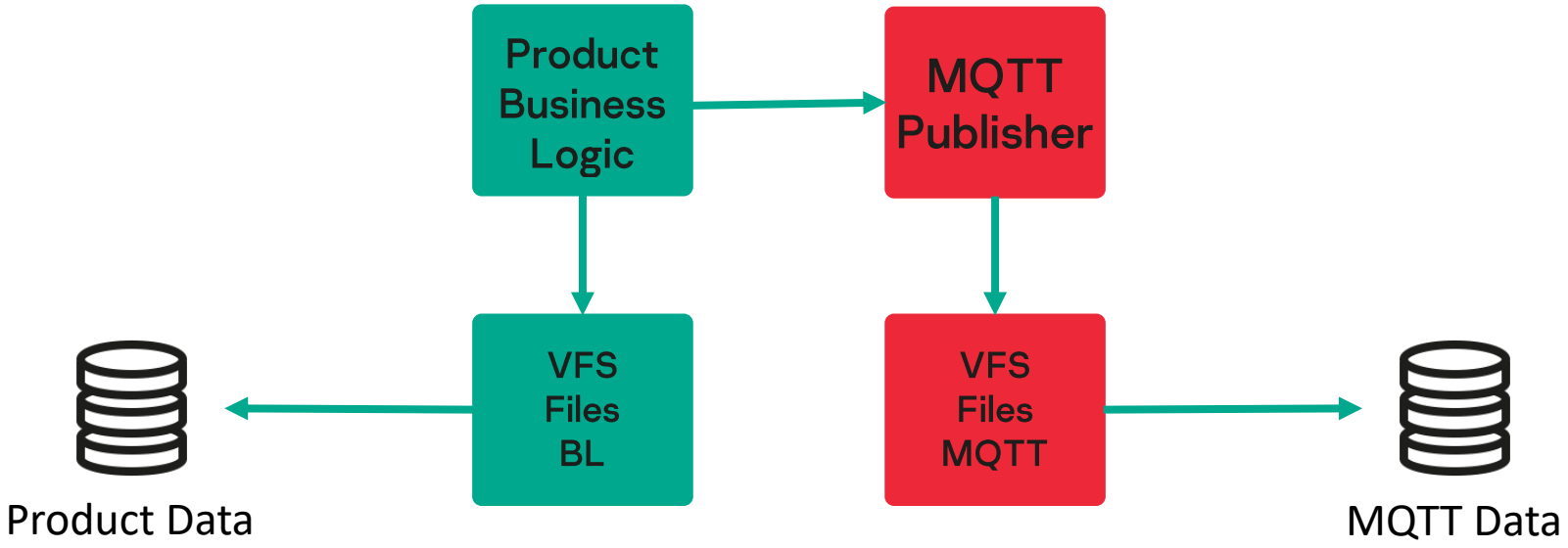
Используем паттерны безопасной архитектуры

1. Разделение сетевой и файловой подсистем



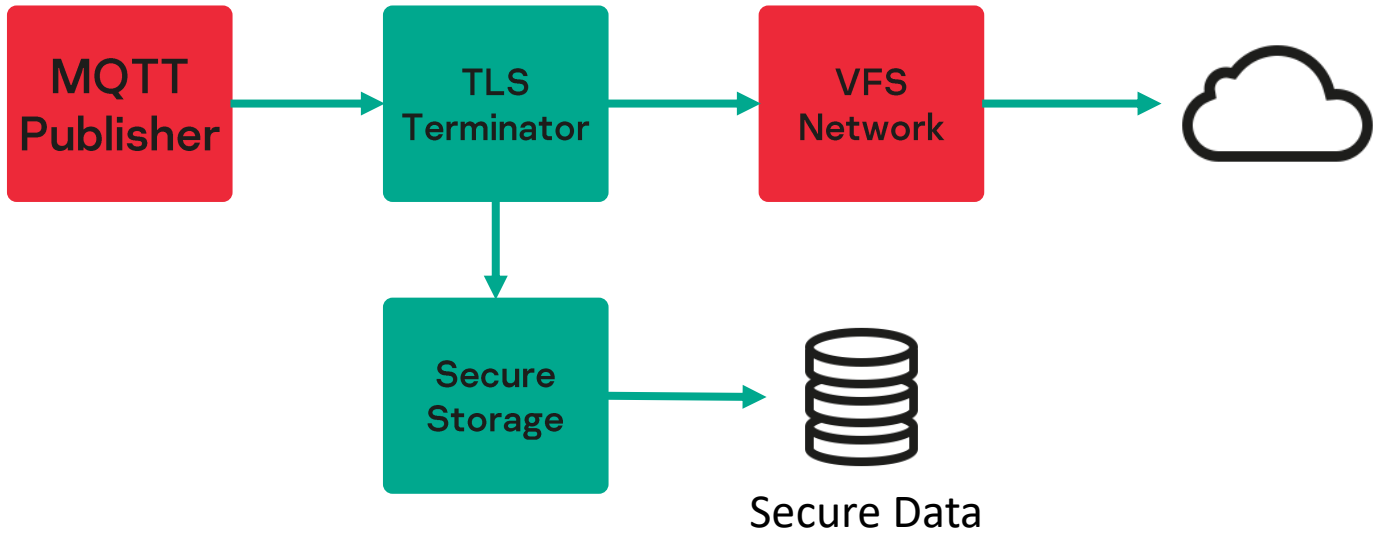
Используем паттерны безопасной архитектуры

2. Разделение файловых систем

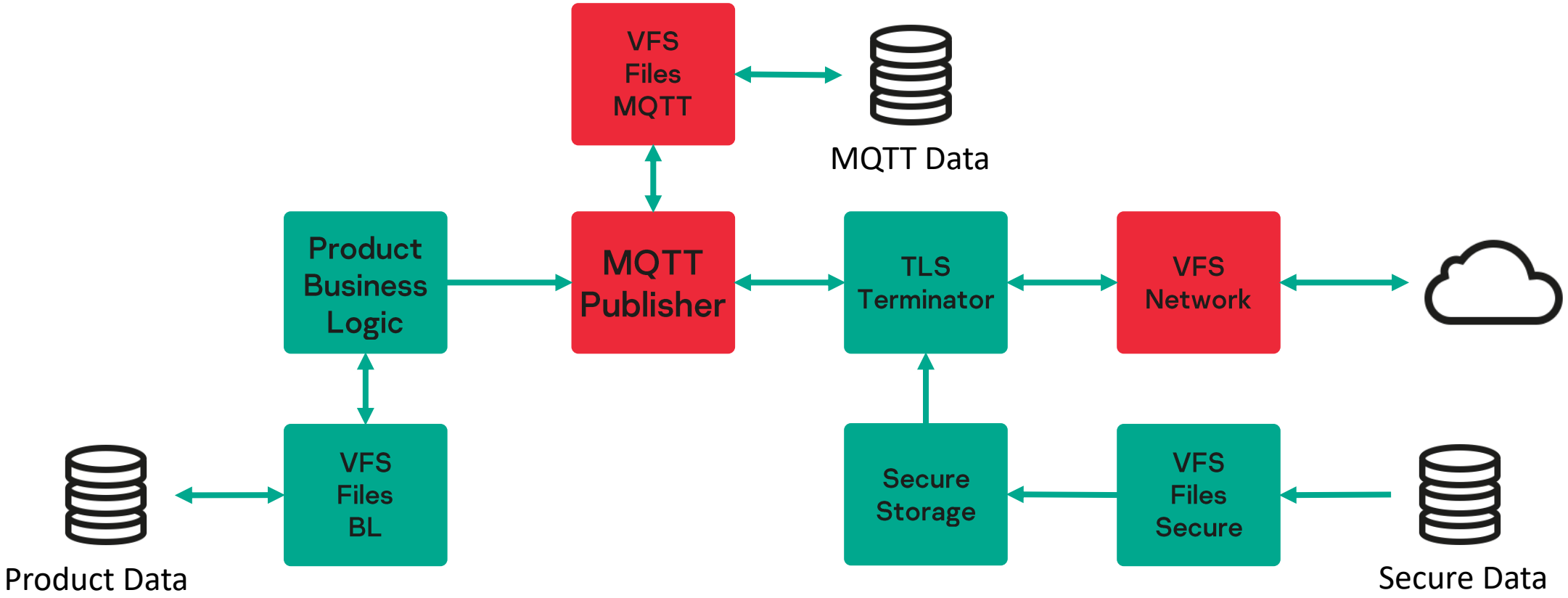


Используем паттерны безопасной архитектуры

3. TLS terminator



Все вместе – диаграмма потоков данных



*Общие компоненты: VFS files, VFS network, TLS Terminator – входят в комплект KasperskyOS SDK и не требуют самостоятельной разработки

Но это еще не все. Дальнейшие шаги:

Но это еще не все. Дальнейшие шаги:

1. Анализ сценариев выполнения целей безопасности

Но это еще не все. Дальнейшие шаги:

1. Анализ сценариев выполнения целей безопасности
2. Имплементация

Но это еще не все. Дальнейшие шаги:

1. Анализ сценариев выполнения целей безопасности
2. Имплементация
3. Написание политик безопасности (контракт взаимодействия элементов системы)

Но это еще не все. Дальнейшие шаги:

1. Анализ сценариев выполнения целей безопасности
2. Имплементация
3. Написание политик безопасности (контракт взаимодействия элементов системы)
4. Подтверждение доверия к ключевым компонентам: тестирование, фаззинг, статический, динамический анализ

Но это еще не все. Дальнейшие шаги:

1. Анализ сценариев выполнения целей безопасности
2. Имплементация
3. Написание политик безопасности (контракт взаимодействия элементов системы)
4. Подтверждение доверия к ключевым компонентам: тестирование, фаззинг, статический, динамический анализ
5. Тесты на проникновение

Но это еще не все. Дальнейшие шаги:

1. Анализ сценариев выполнения целей безопасности
2. Имплементация
3. Написание политик безопасности (контракт взаимодействия элементов системы)
4. Подтверждение доверия к ключевым компонентам: тестирование, фаззинг, статический, динамический анализ
5. Тесты на проникновение
6. **Релиз**

Проверим митигацию угроз по STRIDE

S

T

R

I

D

E

Проверим митигацию угроз по STRIDE

	Угроза	Комментарий
S	Spoofing identify (спуфинг)	Доступ к конфиденциальным данным только из доверенного кода. Надежность аутентификации гарантируется TLS.
T		
R		
I		
D		
E		

Проверим митигацию угроз по STRIDE

	Угроза	Комментарий
S	Spoofing identify (спуфинг)	Доступ к конфиденциальным данным только из доверенного кода. Надежность аутентификации гарантируется TLS.
T		
R		
I		
D		
E		

Проверим митигацию угроз по STRIDE

	Угроза	Комментарий
S	Spoofing identify (спуфинг)	Доступ к конфиденциальным данным только из доверенного кода. Надежность аутентификации гарантируется TLS.
T	Tampering with data (незаконное изменение)	В TLS канале подмена невозможна. Подмена в недоверенном коде теоретически возможна, но практически мало вероятна (необходима дешифрация TLS).
R		
I		
D		
E		

Проверим митигацию угроз по STRIDE

	Угроза	Комментарий
S	Spoofing identify (спуфинг)	Доступ к конфиденциальным данным только из доверенного кода. Надежность аутентификации гарантируется TLS.
T	Tampering with data (незаконное изменение)	В TLS канале подмена невозможна. Подмена в недоверенном коде теоретически возможна, но практически мало вероятна (необходима дешифрация TLS).
R		
I		
D		
E		

Проверим митигацию угроз по STRIDE

	Угроза	Комментарий
S	Spoofing identify (спуфинг)	Доступ к конфиденциальным данным только из доверенного кода. Надежность аутентификации гарантируется TLS.
T	Tampering with data (незаконное изменение)	В TLS канале подмена невозможна. Подмена в недоверенном коде теоретически возможна, но практически мало вероятна (необходима дешифрация TLS).
R	Repudiation (отказ)	В данном примере не рассматривается. В целом возможно.
I		
D		
E		

Проверим митигацию угроз по STRIDE

	Угроза	Комментарий
S	Spoofing identify (спуфинг)	Доступ к конфиденциальным данным только из доверенного кода. Надежность аутентификации гарантируется TLS.
T	Tampering with data (незаконное изменение)	В TLS канале подмена невозможна. Подмена в недоверенном коде теоретически возможна, но практически мало вероятна (необходима дешифрация TLS).
R	Repudiation (отказ)	В данном примере не рассматривается. В целом возможно.
I		
D		
E		

Проверим митигацию угроз по STRIDE

	Угроза	Комментарий
S	Spoofing identify (спуфинг)	Доступ к конфиденциальным данным только из доверенного кода. Надежность аутентификации гарантируется TLS.
T	Tampering with data (незаконное изменение)	В TLS канале подмена невозможна. Подмена в недоверенном коде теоретически возможна, но практически мало вероятна (необходима дешифрация TLS).
R	Repudiation (отказ)	В данном примере не рассматривается. В целом возможно.
I	Information disclosure (раскрытие информации)	Доступ к конфиденциальным данным только из доверенного кода. Раскрытие в TLS канале невозможно.
D		
E		

Проверим митигацию угроз по STRIDE

	Угроза	Комментарий
S	Spoofing identify (спуфинг)	Доступ к конфиденциальным данным только из доверенного кода. Надежность аутентификации гарантируется TLS.
T	Tampering with data (незаконное изменение)	В TLS канале подмена невозможна. Подмена в недоверенном коде теоретически возможна, но практически мало вероятна (необходима дешифрация TLS).
R	Repudiation (отказ)	В данном примере не рассматривается. В целом возможно.
I	Information disclosure (раскрытие информации)	Доступ к конфиденциальным данным только из доверенного кода. Раскрытие в TLS канале невозможно.
D		
E		

Проверим митигацию угроз по STRIDE

	Угроза	Комментарий
S	Spoofing identify (спуфинг)	Доступ к конфиденциальным данным только из доверенного кода. Надежность аутентификации гарантируется TLS.
T	Tampering with data (незаконное изменение)	В TLS канале подмена невозможна. Подмена в недоверенном коде теоретически возможна, но практически мало вероятна (необходима дешифрация TLS).
R	Repudiation (отказ)	В данном примере не рассматривается. В целом возможно.
I	Information disclosure (раскрытие информации)	Доступ к конфиденциальным данным только из доверенного кода. Раскрытие в TLS канале невозможно.
D	Denial of service (отказ в обслуживании)	Пока не защищаем.
E		

Проверим митигацию угроз по STRIDE

	Угроза	Комментарий
S	Spoofing identify (спуфинг)	Доступ к конфиденциальным данным только из доверенного кода. Надежность аутентификации гарантируется TLS.
T	Tampering with data (незаконное изменение)	В TLS канале подмена невозможна. Подмена в недоверенном коде теоретически возможна, но практически мало вероятна (необходима дешифрация TLS).
R	Repudiation (отказ)	В данном примере не рассматривается. В целом возможно.
I	Information disclosure (раскрытие информации)	Доступ к конфиденциальным данным только из доверенного кода. Раскрытие в TLS канале невозможно.
D	Denial of service (отказ в обслуживании)	Пока не защищаем.
E		

Проверим митигацию угроз по STRIDE

	Угроза	Комментарий
S	Spoofing identify (спуфинг)	Доступ к конфиденциальным данным только из доверенного кода. Надежность аутентификации гарантируется TLS.
T	Tampering with data (незаконное изменение)	В TLS канале подмена невозможна. Подмена в недоверенном коде теоретически возможна, но практически мало вероятна (необходима дешифрация TLS).
R	Repudiation (отказ)	В данном примере не рассматривается. В целом возможно.
I	Information disclosure (раскрытие информации)	Доступ к конфиденциальным данным только из доверенного кода. Раскрытие в TLS канале невозможно.
D	Denial of service (отказ в обслуживании)	Пока не защищаем.
E	Elevation of privilege (повышение привилегий)	Уровень привилегий контролируется политиками безопасности.

Проверим митигацию угроз по STRIDE

	Угроза	Комментарий
S	Spoofing identify (спуфинг)	Доступ к конфиденциальным данным только из доверенного кода. Надежность аутентификации гарантируется TLS.
T	Tampering with data (незаконное изменение)	В TLS канале подмена невозможна. Подмена в недоверенном коде теоретически возможна, но практически мало вероятна (необходима дешифрация TLS).
R	Repudiation (отказ)	В данном примере не рассматривается. В целом возможно.
I	Information disclosure (раскрытие информации)	Доступ к конфиденциальным данным только из доверенного кода. Раскрытие в TLS канале невозможно.
D	Denial of service (отказ в обслуживании)	Пока не защищаем.
E	Elevation of privilege (повышение привилегий)	Уровень привилегий контролируется политиками безопасности.

Резюме

Уязвимостей и угроз много, их надо митигировать

Соблюдение процесса SDL позволяет выявлять уязвимости на самых ранних этапах

Подход “Secure by Design” позволяет создавать исходно безопасное ПО

Продукты, обладающие кибериммунитетом, защищены от текущих и будущих угроз

Источники

1. Secure Coding in C and C++, 2nd Edition, Robert C. Seacord
2. Secure Programming Cookbook for C and C++, John Viega, Matt Messier
3. Secure By Design, Daniel Deogun, Dan Bergh Johnsson, Daniel Sawano
4. Technical Guide Security Design Patterns by Bob Blakley, Craig Heath, and members of The Open Group Security Forum The Open Group April 2004
5. CERT secure coding C++
6. CERT secure coding C
7. High integrity C++ coding standard является комбинацией MISRA C++ и JSF AV C++
8. Microsoft SDL
9. Safecode security champion
10. Unusual Bugs, Ilja van Sprundel
11. Threat Modeling: 12 Available Methods
12. KasperskyOS
13. Analyze Crashes to Find Security Vulnerabilities in Your Apps
14. A Beginners Guide To The STRIDE Security Threat Model

Спасибо!

Мы нанимаем!

Сергей Талантов Архитектор ПО sergey.talantov@kaspersky.com

KasperskyOS