OS DAY 2022

Устройство безопасности в KasperskyOS

Как мы делали иммунный RDP-клиент

Сергей Яковлев
Руководитель группы разработки решений для защиты рабочих станций

KasperskyOS

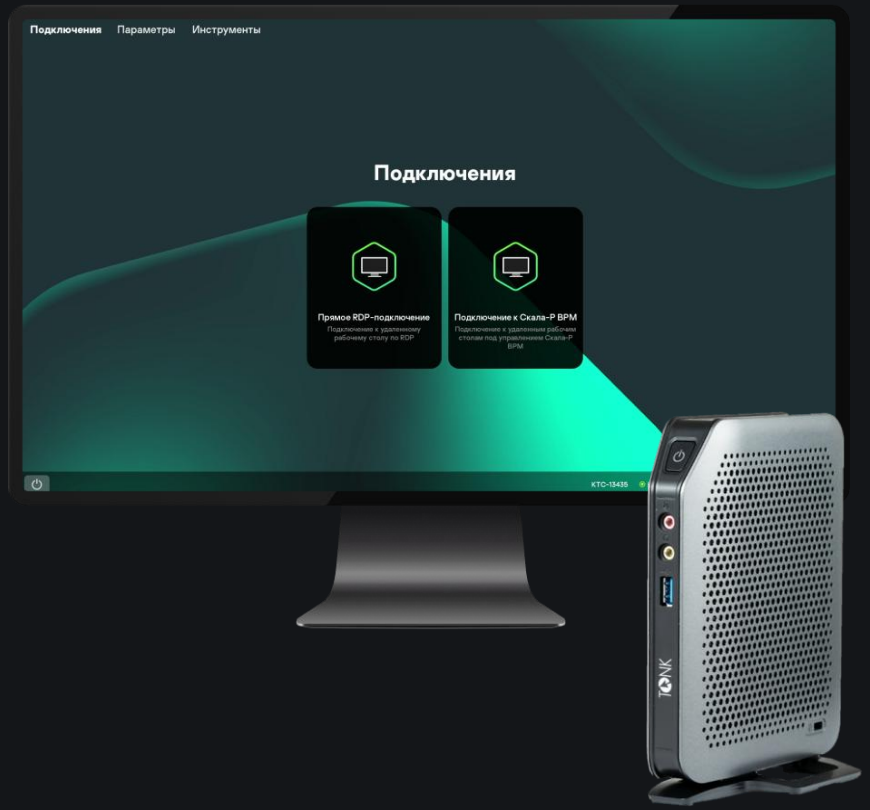
Немного обо мне

3 года в «Лаборатории Касперского»

В проекте с момента его рождения

Начинал с позиции ведущего разработчика

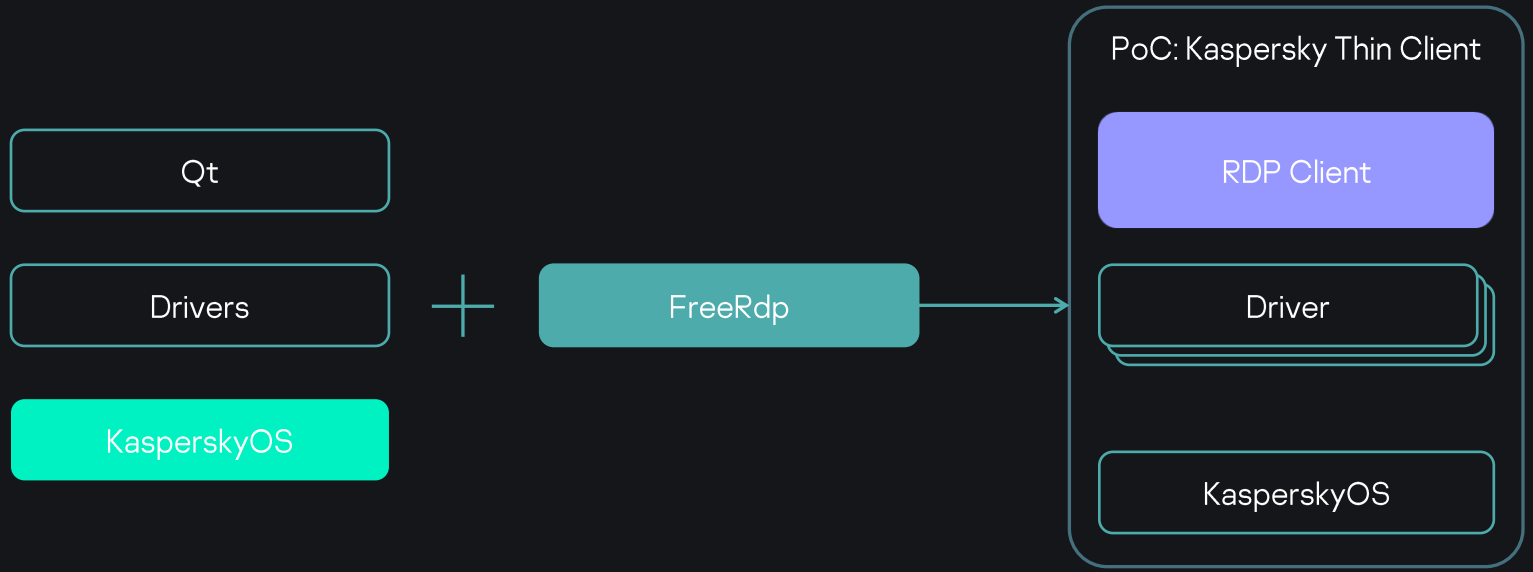
О проекте



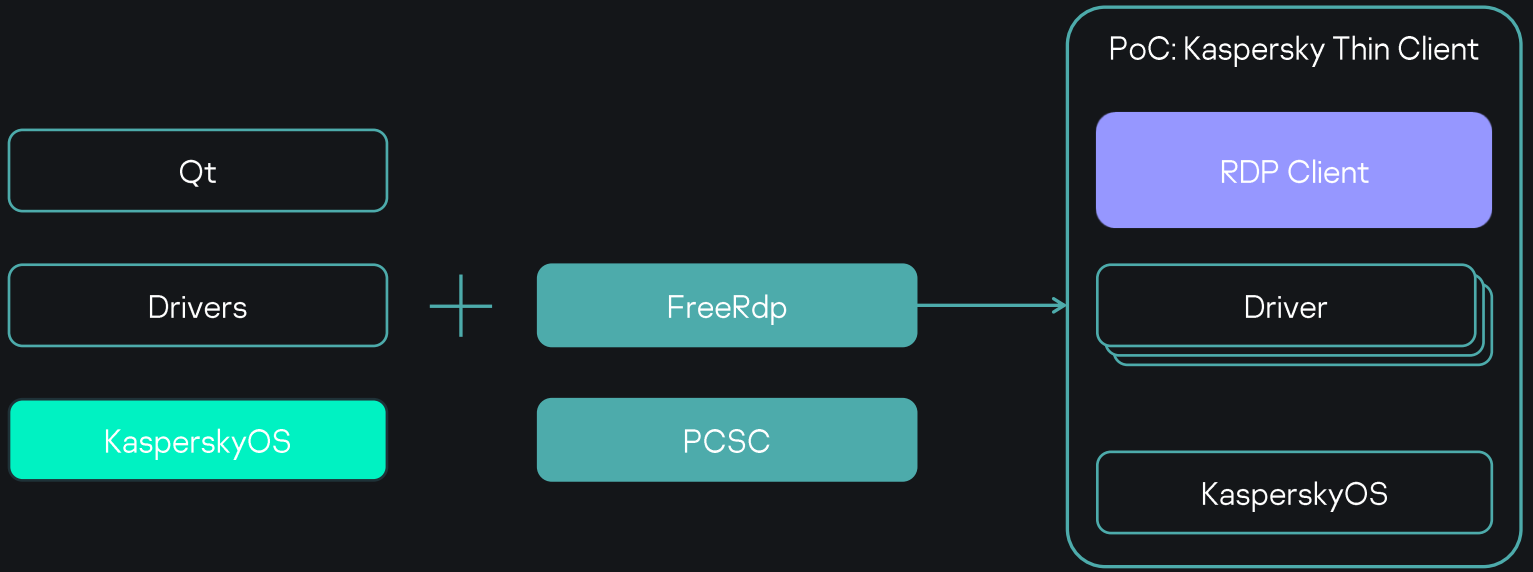
Kaspersky Thin Client — один из основных компонентов инфраструктуры VDI (Virtual Desktop Infrastructure). Обеспечивает безопасное подключение к удаленным рабочим столам по протоколу Remote Desktop (RDP).

FreeRdp — open source реализация клиента и сервера RDP

Первый PoC: KasperskyOS + FreeRdp



PCSC — обеспечивает поддержку проброса смарт-карт в удаленную сессию



Портировать FreeRdp на KasperskyOS просто


```
39
40  #ifdef __APPLE__
41  #include <malloc/malloc.h>
42- #elif __FreeBSD__ || __OpenBSD__
43  #include <stdlib.h>
44  #else
45  #include <malloc.h>
```

```
39
40  #ifdef __APPLE__
41  #include <malloc/malloc.h>
42+ #elif defined(__FreeBSD__) || defined(__OpenBSD__) || defined(__KOS__)
43  #include <stdlib.h>
44  #else
45  #include <malloc.h>
```

Добавляем новую платформу

```
34 +
35 + #define TIMESPEC_TO_TIMEVAL(tv, ts) {
36 +         (tv)->tv_sec = (ts)->tv_sec;
37 +         (tv)->tv_usec = (ts)->tv_nsec / 1000;
38 + }
39 +
```

Добавляем porting layer

```
200     /* fork and exec */
201     pid = fork();
202
203     if (pid < 0)
```

```
200     /* fork and exec */
201 + #ifdef __KOS__
202 +   pid = -1;
203 + #else
204     pid = fork();
205 + #endif
206
207     if (pid < 0)
```

```
94    struct sigaction fatal_sigaction;
95    WLog_DBG(TAG, "Registering signal hook...");
96    sigfillset(&(fatal_sigaction.sa_mask));
97    sigdelset(&(fatal_sigaction.sa_mask), SIGCONT);
98    pthread_sigmask(SIG_BLOCK, &(fatal_sigaction.sa_mask), &orig_set);
99    fatal_sigaction.sa_handler = fatal_handler;
100   fatal_sigaction.sa_flags = 0;
```

```
97    struct sigaction fatal_sigaction;
98    WLog_DBG(TAG, "Registering signal hook...");
99    sigfillset(&(fatal_sigaction.sa_mask));
100 + #ifndef __KOS__
101   sigdelset(&(fatal_sigaction.sa_mask), SIGCONT);
102 + #endif
103   pthread_sigmask(SIG_BLOCK, &(fatal_sigaction.sa_mask), &orig_set);
104   fatal_sigaction.sa_handler = fatal_handler;
105   fatal_sigaction.sa_flags = 0;
```

Заменяем то, что в KasperskyOS отсутствует

```
744
745   status = PCSC_SCardReleaseContext_Internal(hContext);
746
747 - if (status != SCARD_S_SUCCESS)
748       PCSC_ReleaseCardContext(hContext);
749
750   return status;
```

```
744
745   status = PCSC_SCardReleaseContext_Internal(hContext);
746
747 + if (status == SCARD_S_SUCCESS)
748       PCSC_ReleaseCardContext(hContext);
749
750   return status;
```

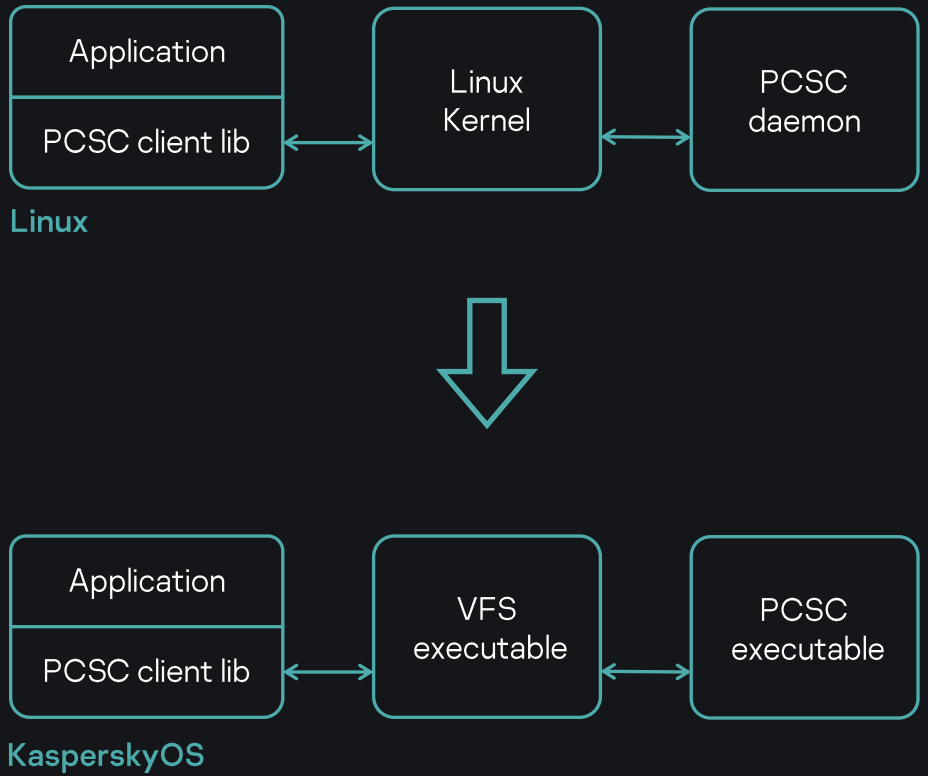
Чиним дефекты

≈30

коммитов

Изменения такие же, как для поддержки любой другой платформы

Портировать PCSC на KasperskyOS еще проще



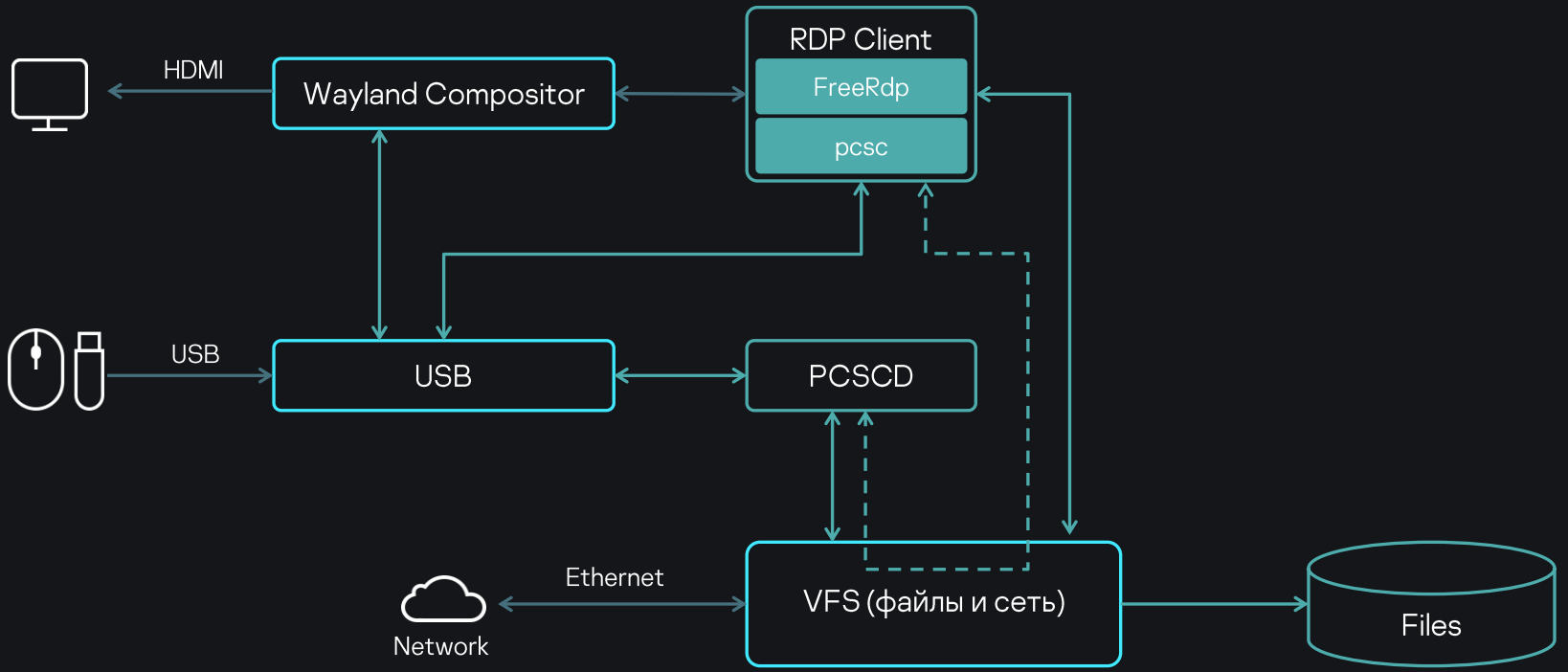
PCSC — демон и клиентская библиотека для работы со смарт-картами

- поддерживается во FreeRdp
- API построен на сокетах

Особенности портирования:

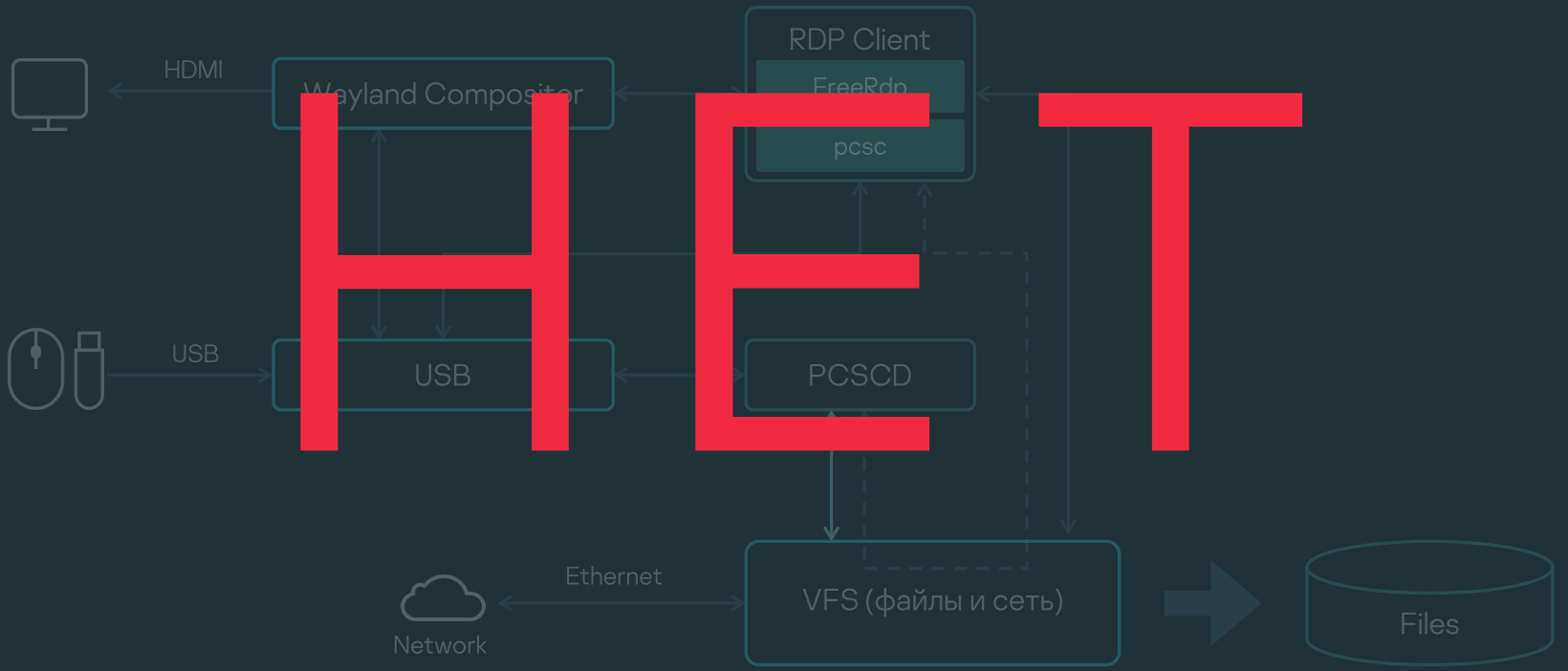
- демон собирается как отдельный исполняемый файл в рамках KasperskyOS
- оставляем API как есть (не нужно разрабатывать специальный IPC)
- минимальные изменения исходного кода и скриптов компонента

Архитектура прототипа RDP-клиента



Является ли такое решение киберимунным?

Архитектура прототипа RDP-клиента



Атака	Возможные последствия
Man In The Middle (MITM)	Злоумышленник получает полный доступ к тому, что делает пользователь

Атака	Возможные последствия
Man In The Middle (MITM)	Злоумышленник получает полный доступ к тому, что делает пользователь
Подмена RDP сервера	Злоумышленник может «увести» учетную запись пользователя

Атака	Возможные последствия
Man In The Middle (MITM)	Злоумышленник получает полный доступ к тому, что делает пользователь
Подмена RDP сервера	Злоумышленник может «увести» учетную запись пользователя
Встраивание и выполнение кода (RCE)	Широкий спектр действий: кража данных, первый шаг к осуществлению других атак и т.д.

Атака	Возможные последствия
Man In The Middle (MITM)	Злоумышленник получает полный доступ к тому, что делает пользователь
Подмена RDP сервера	Злоумышленник может «увести» учетную запись пользователя
Встраивание и выполнение кода (RCE)	Широкий спектр действий: кража данных, первый шаг к осуществлению других атак и т.д.
Физический доступ к ТК в обход ОС	«Покопаться на диске, чтобы найти что-то полезное для злоумышленника»

Требования безопасности RDP-клиента

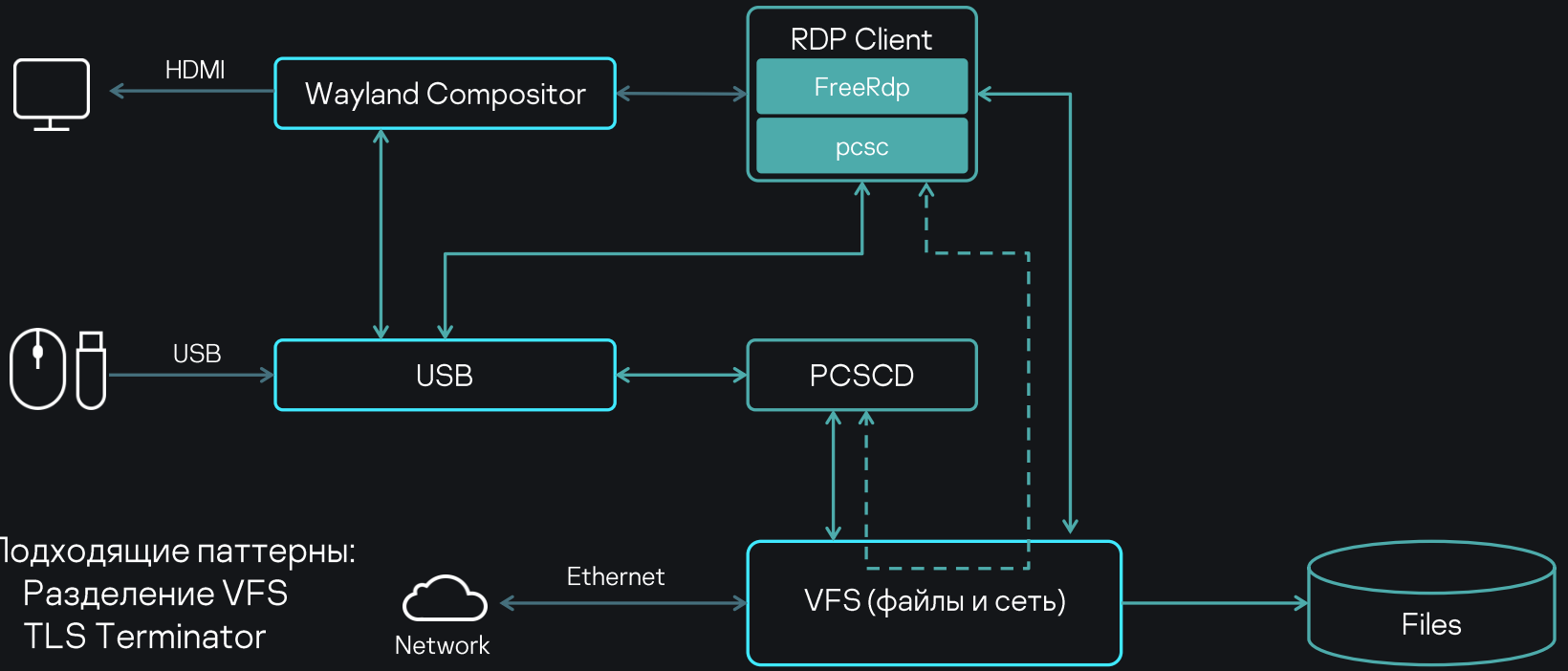
Цели безопасности:

- Целостность данных, полученных от пользователя, непосредственно работающего с устройством
- Конфиденциальность и целостность данных, передаваемых между RDP-клиентом и удаленным рабочим столом

Предположения безопасности:

- У злоумышленника нет физического доступа к ТК

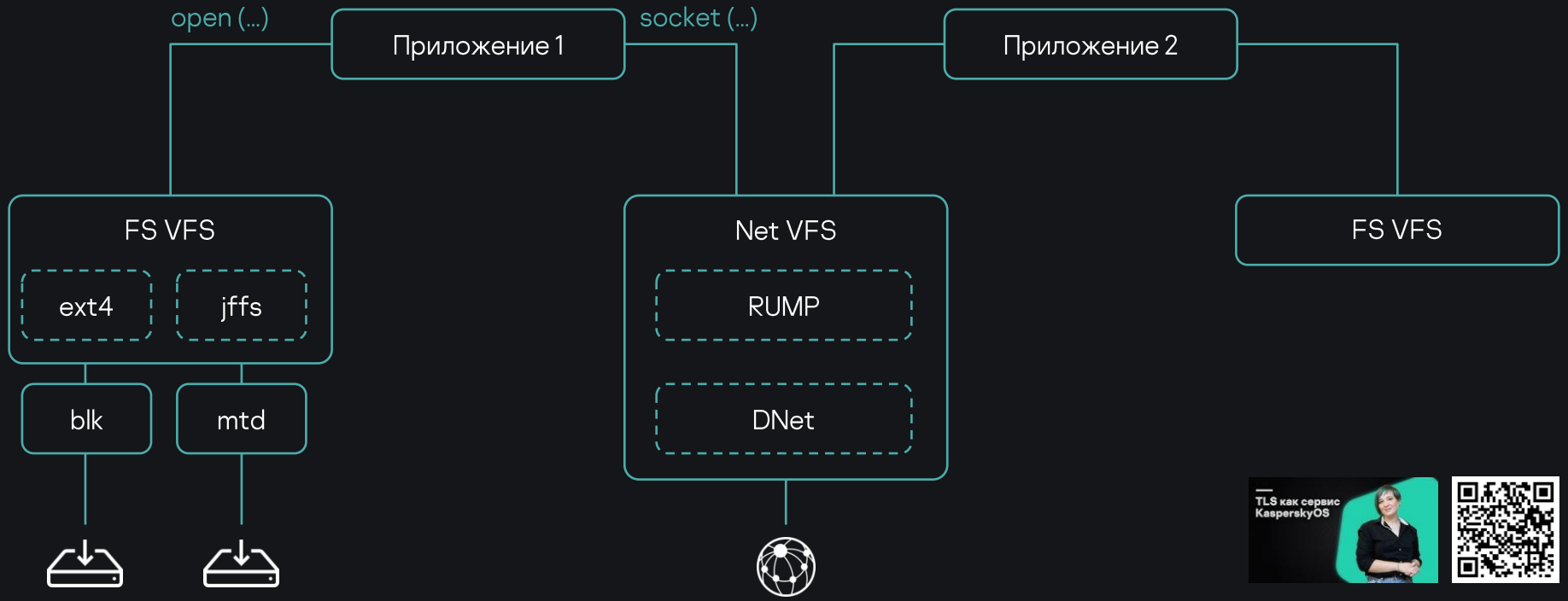
Secure by design: начинаем с архитектуры



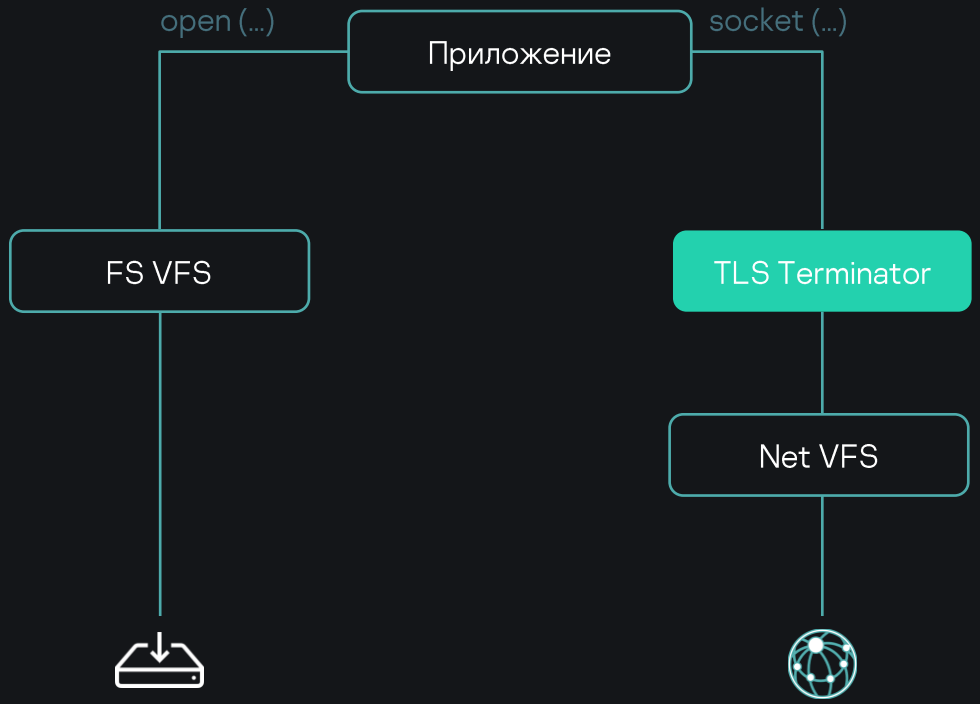
Подходящие паттерны:

- Разделение VFS
- TLS Terminator

Что такое разделение VFS?



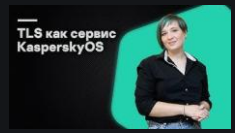
Что такое TLS Terminator?



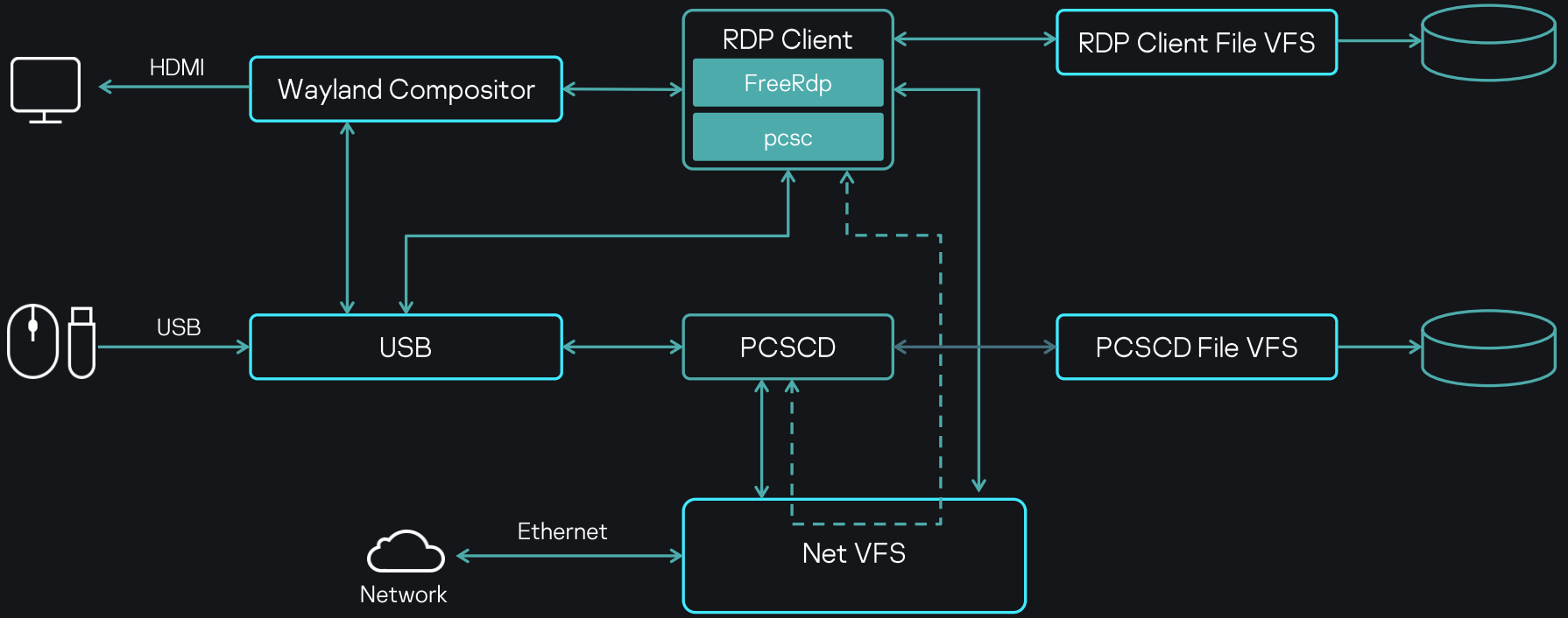
Инкапсулирует шифрование канала передачи данных по сети

Маленький и простой исполняемый файл, доверенность которой легко доказать

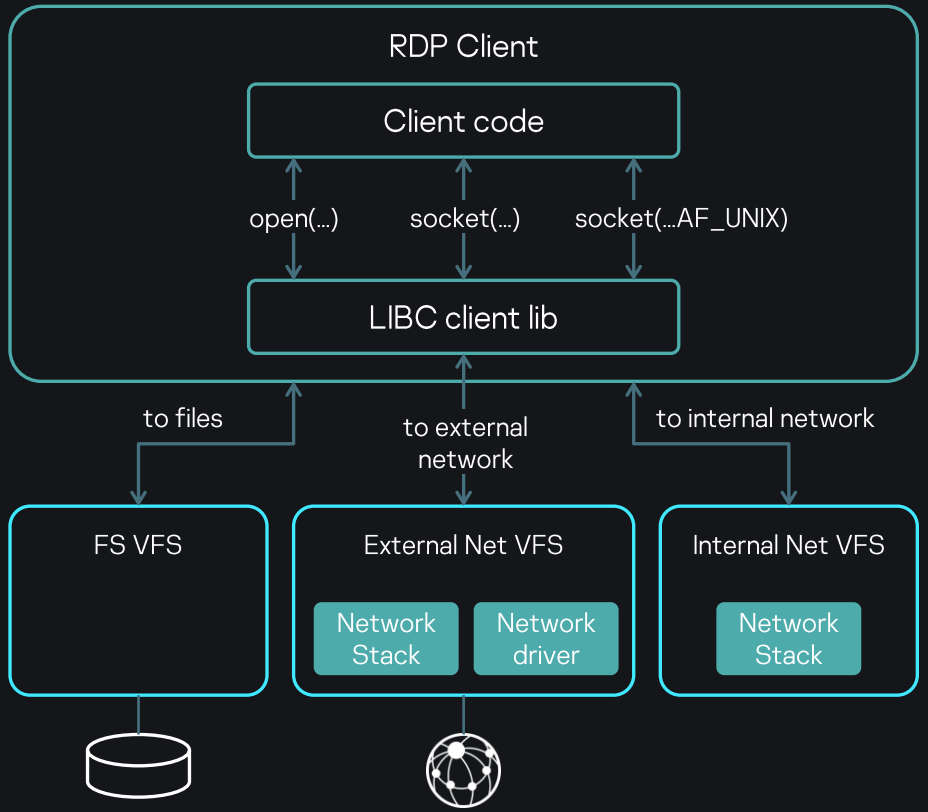
Позволяет защитить клиентов от распространения атаки со стороны сети



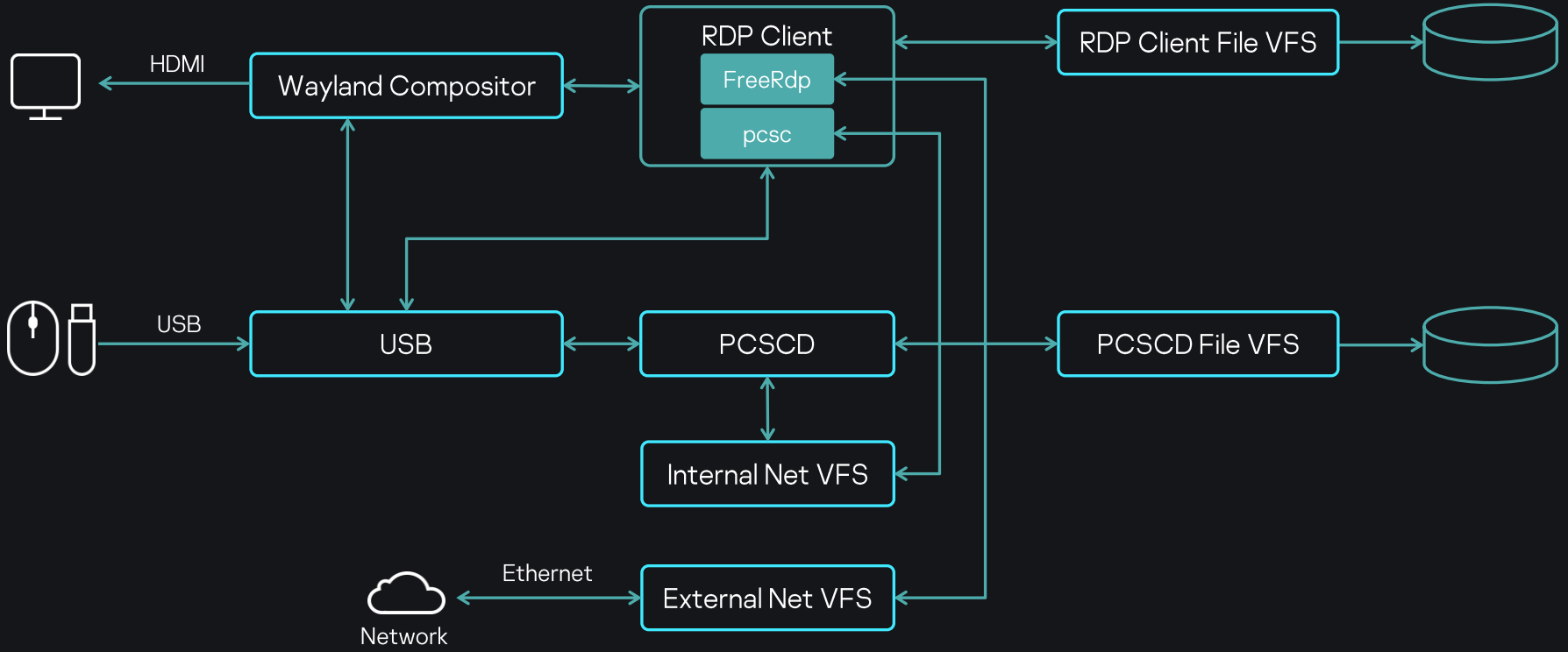
Разделяем потоки данных: файлы отдельно, сокеты отдельно

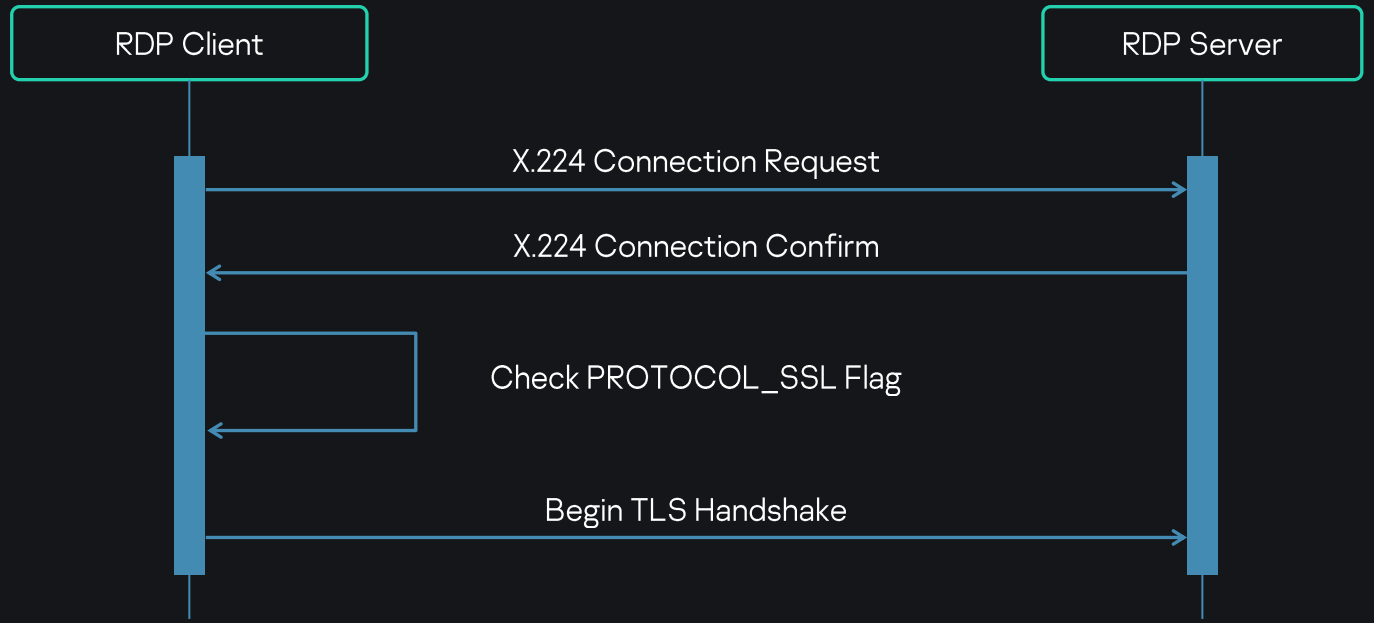


Из коробки не работает — потребовалась доработка клиентской библиотеки VFS

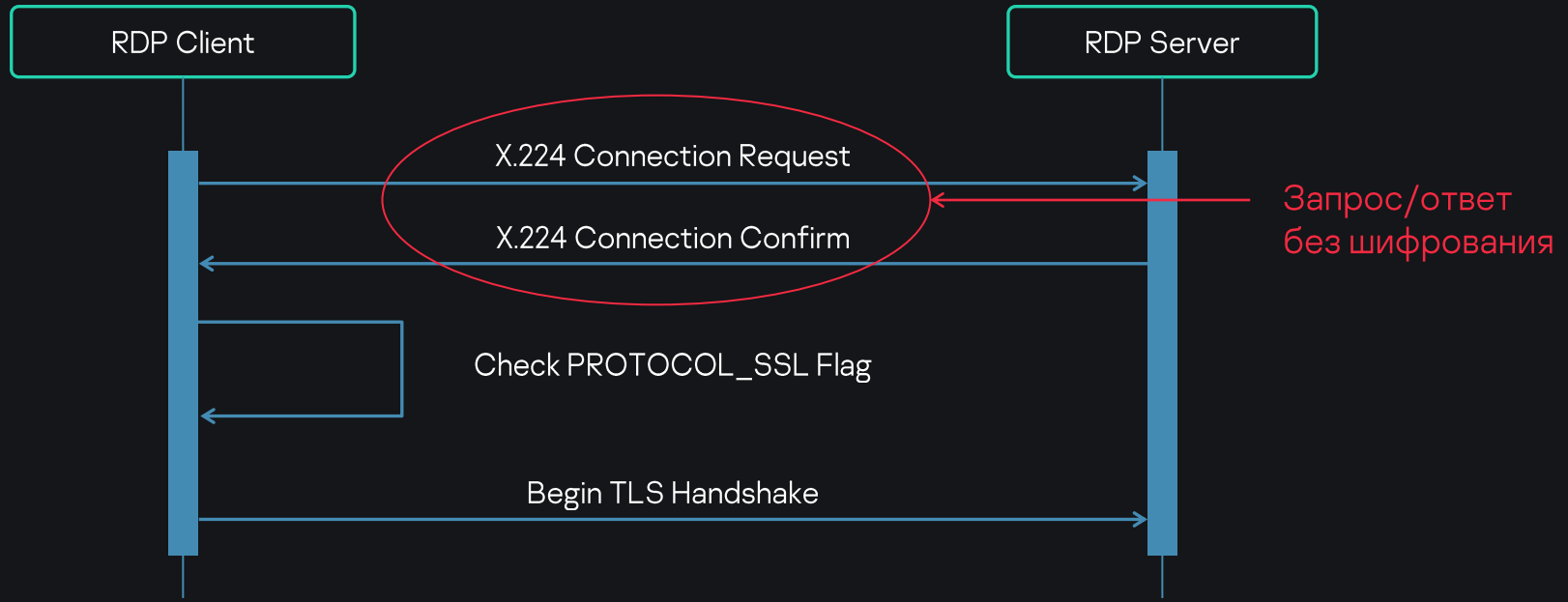


Разделили потоки данных. Что дальше?





Проблема: TLS Terminator неприменим в исходном виде

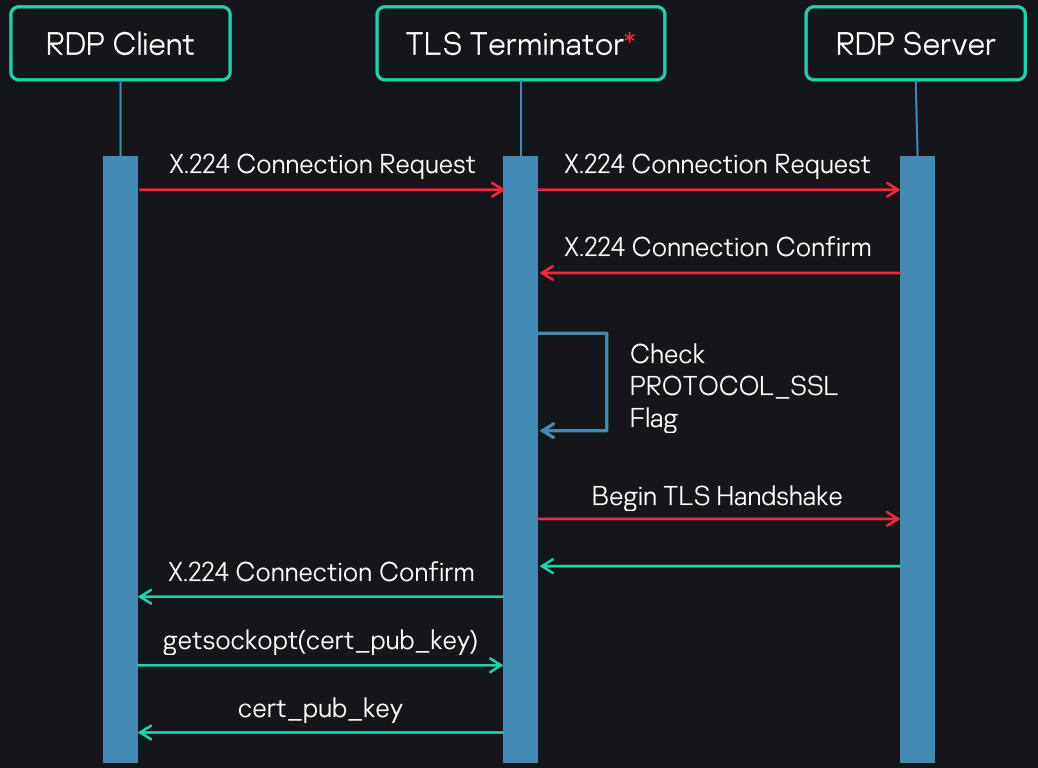


Проблема: TLS Terminator неприменим в исходном виде

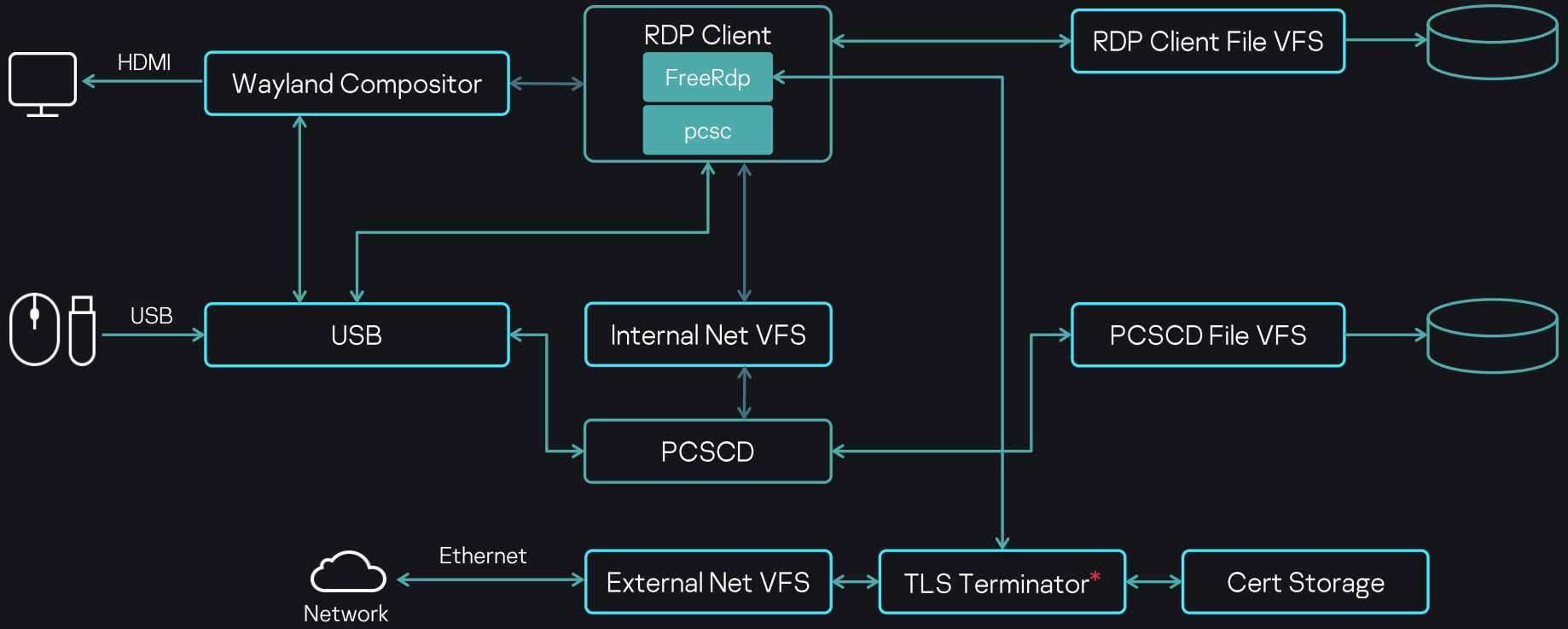
Решение:

Научить TLS Terminator работать с RDP

Просто и быстро, но компонент становится сложнее



Итоговое решение: архитектура



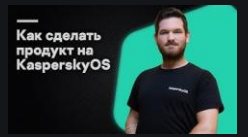
Добавляем политику KSS


```
request src=Pcscd,
        dst=InternalNetVfs,
        endpoint=vfs.server {
    match method=Connect { grant () }
    match method=Pipe { grant () }
    match method=Close { grant () }
    match method=Poll { grant () }
    match method=Socket { grant () }
    match method=Bind { grant () }
    match method=Shutdown { grant () }
    match method=Getsockopt { grant () }
    match method=Read { grant () }
    match method=Recv { grant () }
    match method=Write { grant () }
    match method=Writev { grant () }
    match method=Send { grant () }


    match method=Listen { grant () }
    match method=Accept { grant () }
}
```

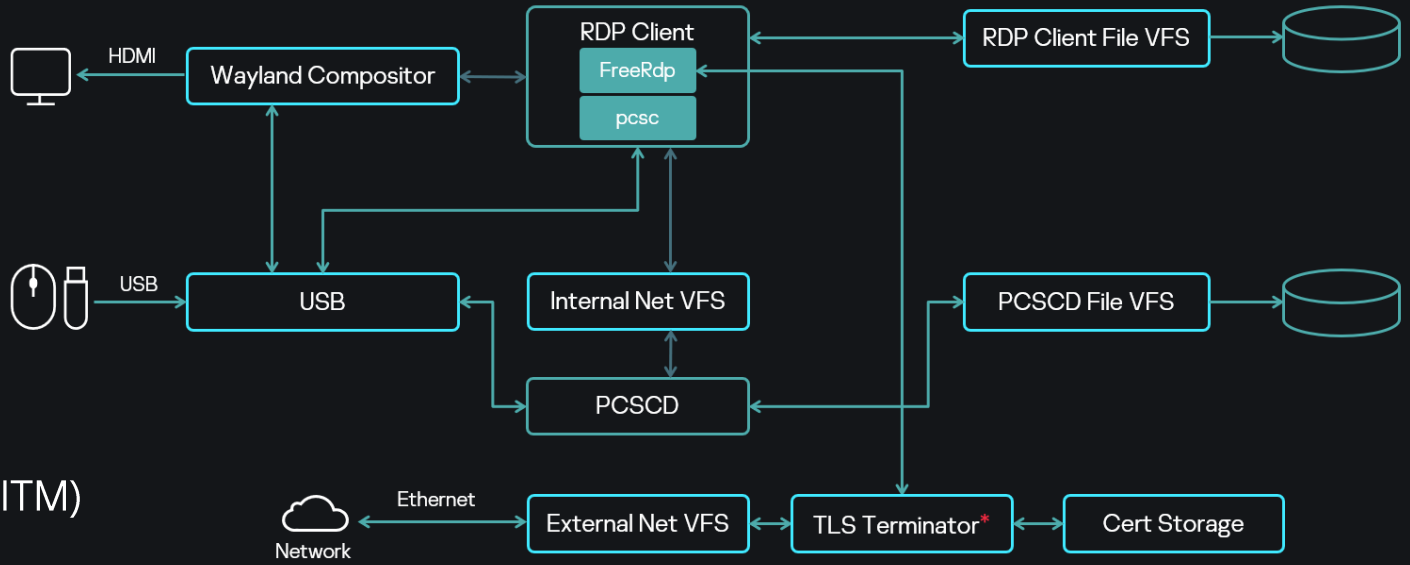
PCSCD -> Internal Net VFS

```
request src=RdpClient,
        dst=InternalNetVfs,
        endpoint=vfs.server {
    match method=Connect { grant () }

    match method=Close { grant () }
    match method=Poll { grant () }
    match method=Socket { grant () }

    match method=Shutdown { grant () }
    match method=Getsockopt { grant () }
    match method=Read { grant () }
    match method=Recv { grant () }
    match method=Write { grant () }
    match method=Writev { grant () }
    match method=Send { grant () }
    match method=Fcntl { grant () }
    match method=Ioctl { grant () }
}
```

RDP Client -> Internal Net VFS



Защищены ли мы теперь?



Man In The Middle (MITM)

Подмена RDP сервера

Встраивание
и выполнение кода (RCE)

Как убедиться, что решение кибериммунно

Метод	Зачем
Построение решения на базе KasperskyOS / KSS	KasperskyOS предоставляет возможность строить кибериммунные решения by design

Как убедиться, что решение кибериммунно

Метод	Зачем
Построение решения на базе KasperskyOS / KSS	KasperskyOS предоставляет возможность строить кибериммунные решения by design
Использование подхода Secure by design	Применяем кибериммунные паттерны при проектировании архитектуры решения

Как убедиться, что решение кибериммунно

Метод	Зачем
Построение решения на базе KasperskyOS / KSS	KasperskyOS предоставляет возможность строить кибериммунные решения by design
Использование подхода Secure by design	Применяем кибериммунные паттерны при проектировании архитектуры решения
Разработка по SDL	Повышаем доверие к коду компонентов. Весь код проходит secure code review, static analysis, dynamic analysis

Как убедиться, что решение кибериммунно

Метод	Зачем
Построение решения на базе KasperskyOS / KSS	KasperskyOS предоставляет возможность строить кибериммунные решения by design
Использование подхода Secure by design	Применяем кибериммунные паттерны при проектировании архитектуры решения
Разработка по SDL	Повышаем доверие к коду компонентов. Весь код проходит secure code review, static analysis, dynamic analysis
Фаззинг-тестирование	Повышаем доверие к компонентам за счет покрытия тестами, в том числе используем фаззинг-тестирование

Как убедиться, что решение кибериммунно

Метод	Зачем
Построение решения на базе KasperskyOS / KSS	KasperskyOS предоставляет возможность строить кибериммунные решения by design
Использование подхода Secure by design	Применяем кибериммунные паттерны при проектировании архитектуры решения
Разработка по SDL	Повышаем доверие к коду компонентов. Весь код проходит secure code review, static analysis, dynamic analysis
Фаззинг-тестирование	Повышаем доверие к компонентам за счет покрытия тестами, в том числе используем фаззинг-тестирование
Пентестирование	Специально обученные инженеры проводят пентестирование для выявления возможных уязвимостей на самом раннем этапе

Какова цена иммунности

Что используем	К чему приводит	Что делать
SDL-практики	Разработка удорожается	Разделяем компоненты на доверенные и недоверенные, основной упор делаем на доверенные компоненты

Какова цена иммунности

Что используем	К чему приводит	Что делать
SDL-практики	Разработка удорожается	Разделяем компоненты на доверенные и недоверенные, основной упор делаем на доверенные компоненты
Новая архитектура RPD-клиента	Потеряли ≈30% FPS	Смогли компенсировать за счет различных оптимизаций в других местах

В KasperskyOS несложно построить кибериммунное решение на базе open source.

В KasperskyOS несложно построить кибериммунное решение на базе open source.

В KasperskyOS реализованы многие паттерны, которые уже можно использовать.

В KasperskyOS несложно построить кибериммунное решение на базе open source.

В KasperskyOS реализованы многие паттерны, которые уже можно использовать.

Кибериммунность не бесплатна, в нашем случае это отразилось на скорости доставки изображения удаленного стола.

В KasperskyOS несложно построить кибериммунное решение на базе open source.

В KasperskyOS реализованы многие паттерны, которые уже можно использовать.

Кибериммунность не бесплатна, в нашем случае это отразилось на скорости доставки изображения удаленного стола.

В итоге мы получаем безопасное решение, способное защитить наших пользователей от основных угроз.

Дополнительные материалы

Kaspersky Thin Client
os.kaspersky.ru/solutions/kaspersky-secure-remote-workspace

FreeRDP
github.com/FreeRDP/FreeRDP

PCSC
github.com/LudovicRousseau/PCSC

RDP
docs.microsoft.com/en-us/openspecs/windows_protocols/ms-rdpbcgr/5073f4ed-1e93-45e1-b039-6e30c385867c

SDL
ru.wikipedia.org/wiki/Security_Development_Lifecycle

TLS Terminator
youtube.com/watch?v=R8isQ9BzhLA

Примите участие в коротком опросе

Спасибо за внимание!

Сергей Яковлев

Руководитель группы разработки решений для защиты рабочих станций

KasperskyOS

sergey.y.yakovlev@kaspersky.com