



# 20 лет проблем приема платежей



## TL;DR

Электронные системы расчетов существуют в интернете уже давно, а баги на них встречаются двадцатилетней давности.

Мы находили критические уязвимости позволяющие угнать деньги и накрутить баланс.

Сегодня мы разберем типовые реализации приема платежей и связанные с ними проблемы безопасности.

## Обзор платежных систем и типовых реализаций API

Мало кто знает, но первой (анонимной!) платежной системой был DigiCash, который появился аж в 1989 году, за ним, в 1996 году, последовала уже более известная (преимущественно среди кардеров) система E-gold.

Но вернемся в настоящее и перечислим основные современные крупные платежные системы/сервисы электронных платежей, которые позволяют принимать платежи на собственном веб-сайте:

- PayPal
- WebMoney
- ЮMoney (бывшие Яндекс.Деньги)
- Qiwi
- Alipay
- и т.д.

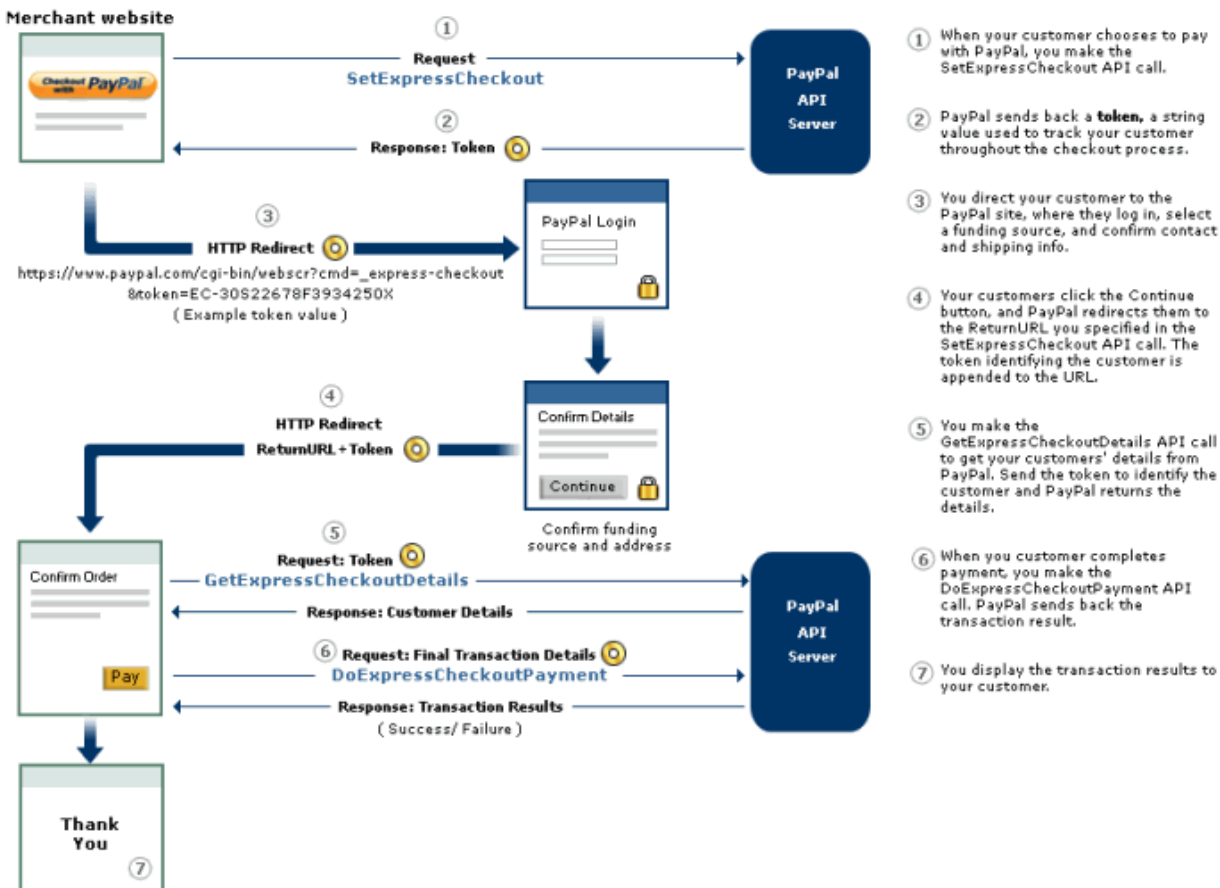


А также десятки менее известных систем, названия которых вам ничего не скажут, не говоря уже о появлении сотен новых, специализирующихся на криптовалютах.

Несмотря на кажущуюся простоту, процесс приема платежей, с точки зрения создания безопасной программной реализации, представляет собой комплексный процесс, который до сих пор приводит к проблемам как у крупных торговых площадок, так и у новых электронных систем расчетов, которые периодически выходят на рынок с "новым и удобным" API и прочими способами интеграции. Как же выглядит типичный процесс приема платежа? Для начала давайте рассмотрим текущую реализацию, которую описывает PayPal, так называемый PayPal Express Checkout.

## PayPal Express Checkout

Legend: → Web Flow → API Call ○ Token



Данную реализацию можно считать относительно безопасной и вот почему:

- Параметры платежа не передаются явным образом, вместо этого используется Token
- Сервер платежной системы не отправляет результаты на некий URL самостоятельно, вместо этого ваш веб-сайт должен самостоятельно их запросить и обработать ответ
- В целом схема взаимодействия реализована так, что у потенциального разработчика существует минимум возможностей "выстрелить себе в ногу"

А теперь посмотрим на схему, которую нам предлагает WebMoney.

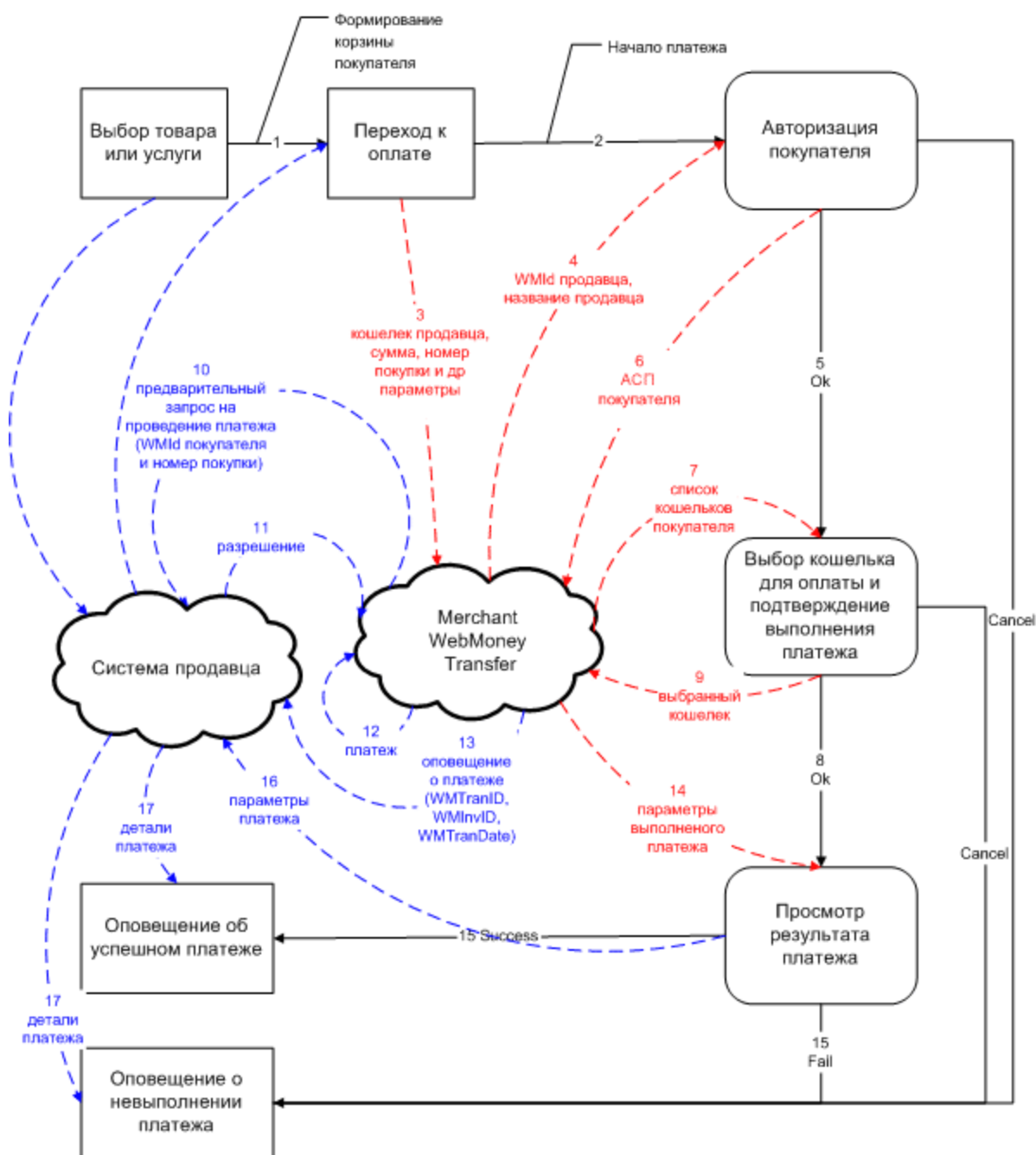


Схема ни хрена не понятная. Также схема не отражает ряд нюансов, вроде подписи запроса. Или информацию о том, что URL, который принимает на себя технические параметры платежа от платежной системы и URL, куда пользователь будет перенаправлен для просмотра деталей об оплате, стоит делать разными. Архитектура, которую использует WebMoney, часто всплывает в той или иной форме и в других платежных системах, которые были созданы в СНГ.

## Типовые проблемы

Излишнее усложнение схемы приема платежей ведет к финансовым потерям. Например, 10 лет назад Kaimi публиковал [заметку](#) о проблеме интеграции с WebMoney системы Global Collect Services, что приводило к возможности подтверждать платежи без оплаты в Steam, [Battle.net](#) и некоторых других.

В чем же состояла проблема? Ранее я упоминал URL на стороне продавца, которые должны принимать информацию о платеже. Согласно [документации](#), у WebMoney существуют три сущности:

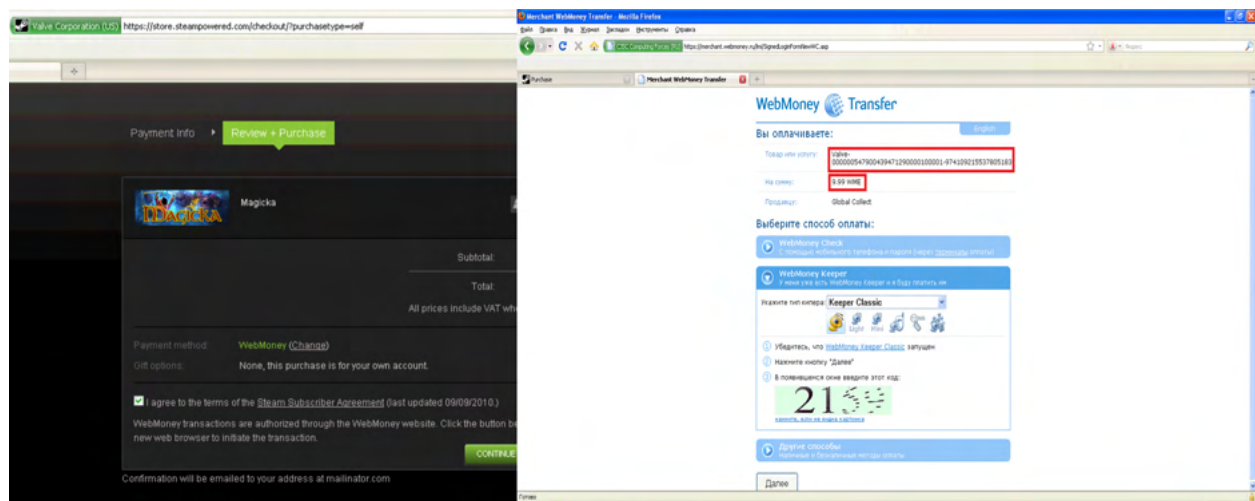
- **Success URL** - URL (на веб-сайте продавца), на который будет переведен интернет-браузер покупателя в случае успешного выполнения платежа в сервисе Web Merchant Interface. URL имеет префикс "http://" или "https://".
- **Fail URL** - URL (на веб-сайте продавца), на который будет переведен интернет-браузер покупателя в том случае, если платеж в сервисе Web Merchant Interface не был выполнен по каким-то причинам. URL имеет префикс "http://" или "https://".
- **Result URL** - URL (на веб-сайте продавца), на который сервис Web Merchant Interface посылает HTTP POST или SMTP-оповещение о совершении платежа с его детальными реквизитами. URL должен начинаться с префикса "http://", "https://" или "mailto:".

Что делают некоторые разработчики, которые читают эту документацию:

1. Обрабатывают информацию по одному URL, что приводит к возможности узнать адрес обработчика (также обработчик, в том числе Result URL, может отображаться в платежной форме на сайте WebMoney, но такое происходит не всегда и, вероятно, зависит от настроек).

2. Некорректно реализуют проверку подписи для запроса, который приходит на Result URL. Это позволяет клиенту подменить данные о платеже.
3. Проверяют подпись, но не проверяют сумму, которая пришла на Result URL. Это позволяет получить товар за 100\$, заплатив, например, 0.01\$.
4. Проверяют подпись, сумму, но не нотацию передаваемых сумм. Помните я упоминал о передаче параметров платежа через браузер клиента? Так вот, WebMoney абсолютно нормально воспринимает сумму платежа со значением **1e1** или **0xFF**, а сравнение подобных чисел, еще и на старых версиях PHP, еще и с учетом нюансов сравнения в языке PHP, приводило к самым неожиданным последствиям.
5. Не совсем проблема платежной системы, НО, как насчет race condition и одинаковых внутренних идентификаторов платежей на ресурсе продавца? Привет, мультипликация баланса.
6. ...

## Подпись запросов

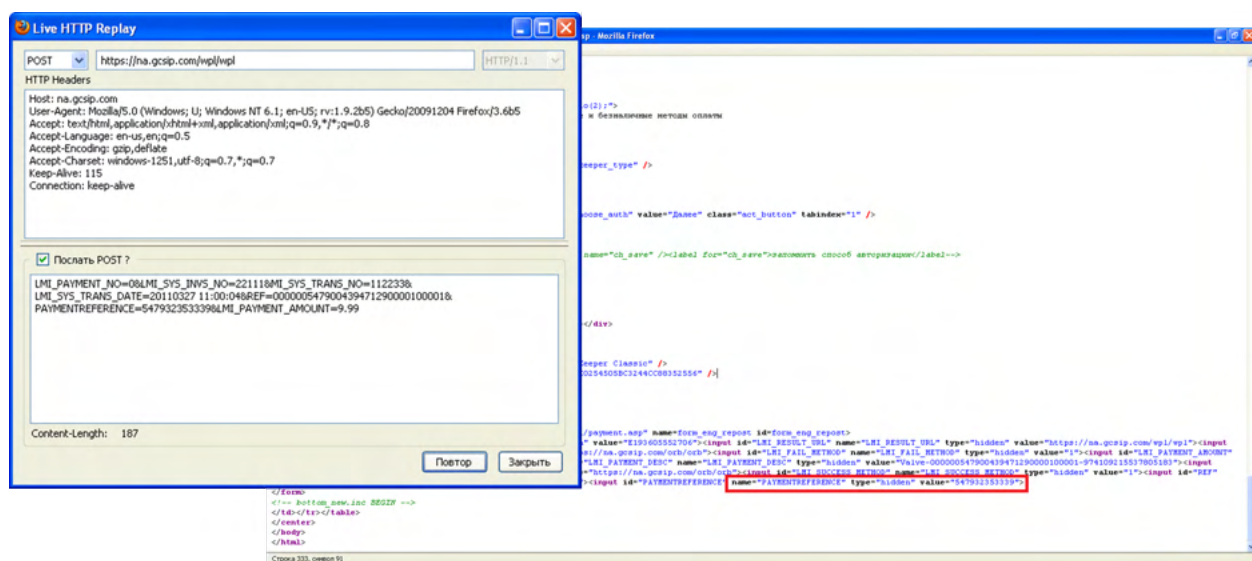


Как это работало:

1. При оплате через WebMoney, пользователя, в соответствии со спецификацией платежной системы, перенаправляло на сайт WebMoney, где он мог видеть сумму платежа, номер счета и прочие параметры.



2. После нажатии кнопки “Далее” и аутентификации в системе становилась доступна информация об URL, который отвечал за обработку результата платежа (Result URL).
3. Пользователь мог сформировать запрос к целевому URL, который, в соответствии со спецификацией WebMoney (ну почти), информировал платежную систему о том, что платеж успешно проведен.
4. Profit!



Система приема платежей Global Collect успешно споткнулась о несколько проблем, которые упоминались выше:

- Известный единый обработчик результатов платежа
- Отсутствие проверки подписи (да и отсутствие подписи в запросе как таковой)
- Использование данных, передаваемых через браузер пользователя, в качестве (хотя, согласно спецификации WebMoney, это можно было делать через коллбэк, приходящий от серверов WebMoney)

Все это привело к возможности совершать фиктивные транзакции и покупать все, что использовало процессинг Global Collect, без ограничений.

Проблему устранили только через ~2 недели массовой эксплуатации.



Другой вариант похожей проблемы, но чуть сложнее, был не так давно в Smart2Pay.

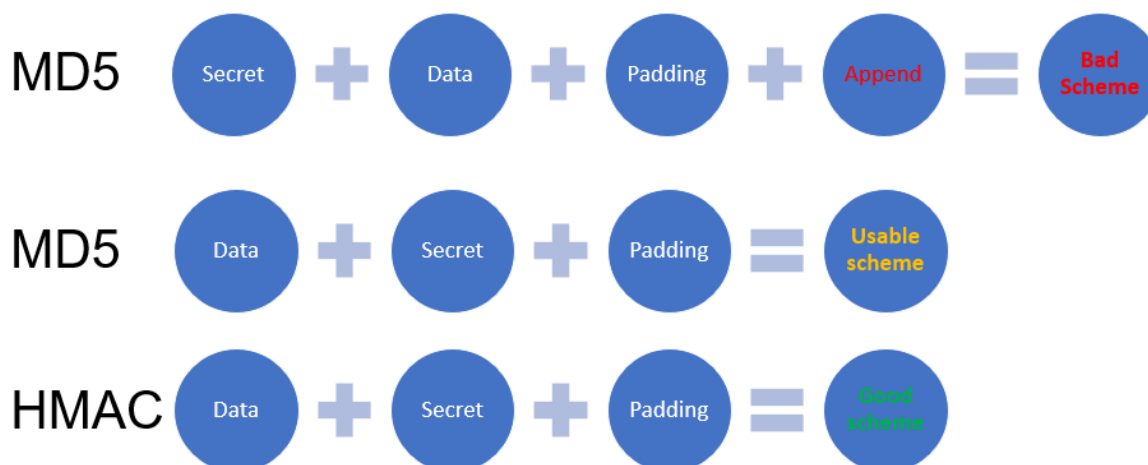
Еще одна проблема, связанная с подписью запросов - **Length Extension Attack**.

- A type of attack against **hash functions** which allow inclusion of extra data without the knowledge of **secret**
- Attack details
  - Knowledge:  $h(s||m)$  and  $m$ ,
  - Target: Appends  $m'$  to  $m$ , and computes correct  $h(s||m||m')$
  - **Exploit**: A vulnerability in Merkle–Damgård construction, which literally calls hash functions on a message block basis.
- References:
  - [http://en.wikipedia.org/wiki/Length\\_extension\\_attack](http://en.wikipedia.org/wiki/Length_extension_attack)

Или атака удлинением сообщения. Согласно Wikipedia - это тип атаки на хеш-функцию, заключающейся в добавлении новой информации в конец исходного сообщения. При этом новое значение хэша может быть вычислено, даже если содержимое исходного сообщения остаётся неизвестным. Чуть подробнее можно изучить здесь.

Проблема встречалась всего пару раз, когда разработчики решили реализовать свою “классную” подпись запросов в стиле VK (которые в общем-то тоже не сами придумали алгоритм), но получилось как обычно.

Ниже небольшая иллюстрация на тему как допустимо генерировать подпись в таком вот стиле и как “выстрелить себе в ногу”.



Для эксплуатации же можно воспользоваться одним из следующих инструментов:

<https://github.com/bwall/HashPump>

[https://github.com/iagox86/hash\\_extender](https://github.com/iagox86/hash_extender)

## Раскрытие “Result URL”

На сайте, где было доступно пополнение с помощью WooPay (через SMS), отображался полный URL с параметрами (включая подпись), по которому платежная система уведомляет сайт, если платеж успешно зачислен.

Логика достаточно проста, нужно вызвать исключительную ситуацию, чтобы тестируемое веб-приложение вывело ошибку.

Если выбрать оплату через SMS и ввести случайный недействительный номер, то получаем:



Повторяем запрос сотню-другую раз. Отправив нас в бан, сайт начал выводить exception, в тексте которого содержался тот самый секретный URL, перейдя по которому на счет зачисляются деньги.

```

nceId";s:14:"6622473.193695";s:7:"backUrl";s:56:"";s:10:
:l";s:94:"?trid=6622473.193695&sign=462371e223dc12a547b228cdd2dea936";s:7:"a
4:"";s:6:"amount";d:193695;s:9:"deathDate";s:19:"2016-03-15
":38";s:11:"description";s:18:"WoopPay ->
123";s:11:"orderNumber";N;s:11:"serviceType";i:2;}}}}i:2;a:6:{s:4:"file";s:42:"/usr/local/www/data-dist/common/w
:s:4:"line";i:133;s:8:"function";s:26:"cash_createInvoiceExtended";s:5:"class";s:10:"SoapClient";s:4:"type";s:2:'
args";a:1:{i:0;r:27;}}i:3;a:6:{s:4:"file";s:45:"/usr/local/www/data-dist/common/index_top.php";s:4:"line";i:1246
:ion";s:7:"Payment";s:5:"class";s:7:"WoopPay";s:4:"type";s:2:"->";s:4:"args";a:3:{i:0;d:193695;i:1;s:7:"1044123"

```

## Атрибуты платежа

Перейдем к проблеме проверки атрибутов платежа.

Один из вариантов интеграции с ЮMoney (ex Яндекс.Деньги), это форма для перевода или ее старая реализация. Опознать ее можно по наличию запросов

<https://yoomoney.ru/eshop.xml>

или

<https://yoomoney.ru/quickpay/confirm.xml>

Прямо при отправке запроса необходимо подменить число, которое запрашивает тестируемое веб-приложение:

The screenshot shows the 'Request' and 'Response' tabs of a web browser's developer tools. The 'Request' tab is active, showing a POST request to `/quickpay/confirm.xml` with a long URL-encoded body. The 'Response' tab is also visible, showing an HTTP 302 Found status and a redirect to a transfer page.

**Request:**

```

1 POST /quickpay/confirm.xml HTTP/1.1
2 Host: yoomoney.ru
3 Content-Length: 321
4 Cache-Control: max-age=0
5 Sec-Ch-Ua: " Not A;Brand";v="99", "Chromium";v="90"
6 Sec-Ch-Ua-Mobile: ?0
7 Upgrade-Insecure-Requests: 1
8 Origin: https://soap4.me
9 Content-Type: application/x-www-form-urlencoded
10 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.212 Safari/537.36
11 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
12 Sec-Fetch-Site: cross-site
13 Sec-Fetch-Mode: navigate
14 Sec-Fetch-User: ?1
15 Sec-Fetch-Dest: document
16 Referer: https://soap4.me/
17 Accept-Encoding: gzip, deflate
18 Accept-Language: en-US,en;q=0.9
19 Connection: close
20
21 receiver=41001477913566formcomment=%D1%80%D0%B0%D1%81%D1%88%D0%B8%D1%80%D0%B5%D0%B0%D0%B0%D1%88%D0%B9+%D0%B0%D0%BA%D0%BA%D0%B0%D1%83%D0%BD%D1%82&label=12pro+45272+16281711996quickpay-form=shop&targets=12pro4527216281711996sum=850&comment=&need-fto=false&need-email=false&need-phone=false&need-address=false&paymentType=AC

```

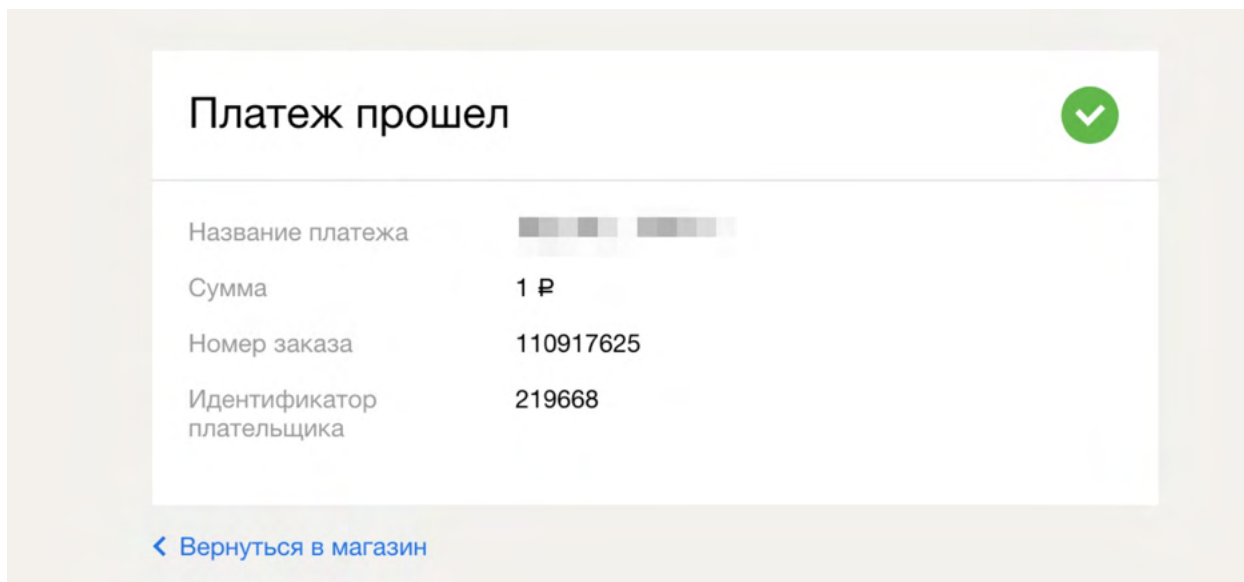
**Response:**

```

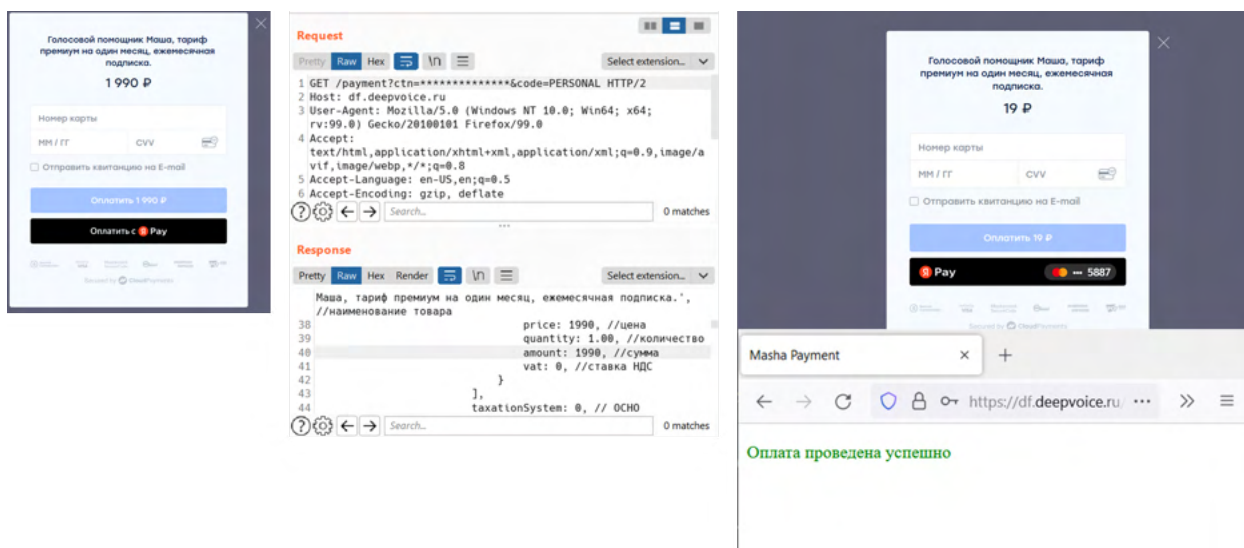
1 HTTP/1.1 302 Found
2 Date: Thu, 05 Aug 2021 13:46:46 GMT
3 Content-Type: text/html; charset=utf-8
4 Content-Length: 344
5 Connection: close
6 Set-Cookie: srv_id=4aedaaaf24359e1b5b287011afa2c87e2; path=/
7 X-Robots-Tag: noindex
8 Location: https://yoomoney.ru/transfer/quickpay?requestId=34363233332363732345f62:361333063636363393663663161656537306238646533343565396233346330643161663:3837
9 Vary: Accept
10 Cache-Control: no-store, must-revalidate, max-age=0
11 Pragma: no-cache
12 Set-Cookie: TS015643f9=01f3111baff543e9577c392502697ecffe85f26e5c5e77275da9a4f4f486d:84ae3ef9ee21bf6323606b05e9dbd6f3ab4463a88993; Path=/
13
14 <p>Found. Redirecting to <a href="https://yoomoney.ru/transfer/quickpay?requestId=34363233332363732:45f6233613330636363393663663161656537306238646533343565396233346330643:6166393837">https://yoomoney.ru/transfer/quickpay?requestId=34363233332:63732345f6233613330636363393663663161656537306238646533343565396233346:3064316166393837</a></p>

```

Высокая вероятность, что платеж пройдет. А дальше сайт либо проверяет сумму, либо нет. Так как в комментарии к платежу передается идентификатор пользователя и/или идентификатор платежа, этого достаточно, чтобы оформить подписку на какой-то сервис (например, на [rbc.ru](https://rbc.ru)).



Небольшой пример:



На скриншоте происходит оплата подписки на бота голосового ассистента в Telegram, но сумма платежа не проверяется, что дает приобрести продукт за произвольную цену. Подменяем сумму 1990 на 19, получаем нужную подписку. Этот тип проблем часто встречается (например, много сервисов-ботов в Telegram, о которых многие слышали и где эта проблема до сих пор присутствует), в том числе на зарубежных ресурсах (пример из старых - покупка лицензии Minecraft), в 2022 году, хотя казалось бы...

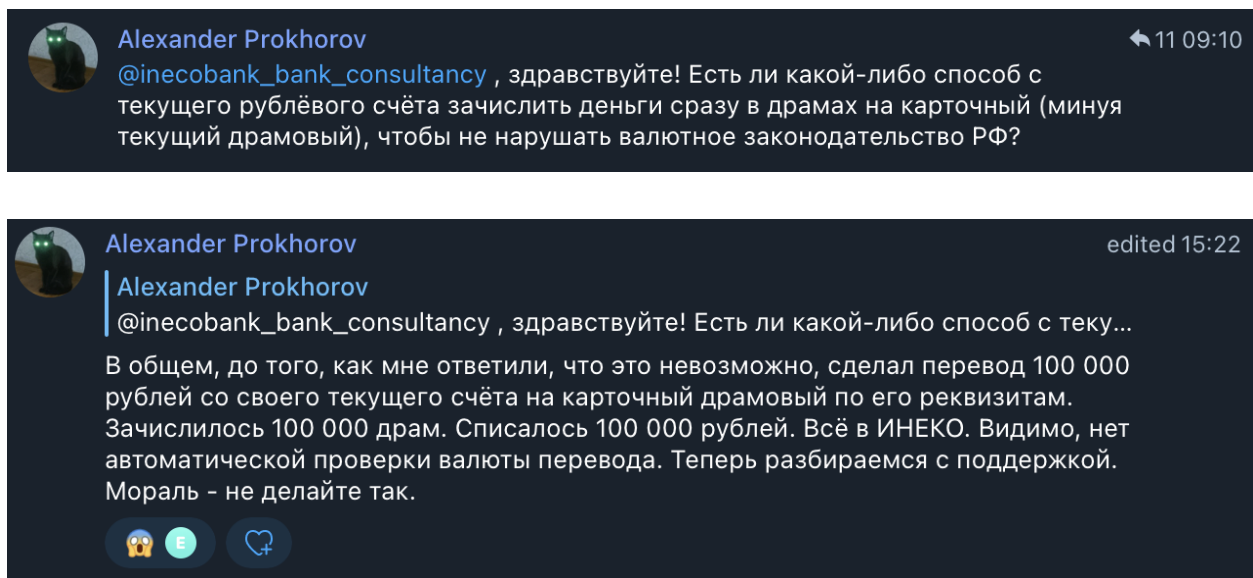
Домашнее задание #1

Найти платный telegram-бот, попробовать оплатить подписку минимальной суммой

Еще один релевантный пример, но связанный не с суммой платежа, а с валютой, присутствовал у QIWI. В кошельке была опция пополнения путем отправки SMS на короткий номер, причем валюта передавалась через браузер клиента на нескольких этапах (выбор валюты и суммы, отправка SMS), где сервер доверял данным клиента. В итоге на счет зачислялось 100\$, а оплата на 100р.

А что там в 2022?

Смотрим в чат армянского банка [https://t.me/Inecobank\\_forum/6333](https://t.me/Inecobank_forum/6333)



Значит проблема до сих пор актуальна.

## Нотации и сравнение типов

#### Example #1 Integer literals

```
<?php
$a = 1234; // decimal number
$a = 0123; // octal number (equivalent)
$a = 0o123; // octal number (as of PHP 7.0)
$a = 0x1A; // hexadecimal number (equivalent)
$a = 0b11111111; // binary number (equivalent)
$a = 1_234_567; // decimal number (as of PHP 7.4)
?>
```

Loose comparisons with ==												
	true	false	1	0	-1	"1"	"0"	"-1"	null	[]	"php"	""
true	true	false	true	false	true	true	false	true	false	false	true	false
false	false	true	false	true	false	false	true	false	true	true	false	true
1	true	false	true	false	false	true	false	false	false	false	false	false
0	false	true	false	true	false	false	true	false	true	false	false	false
-1	true	false	false	false	true	false	false	true	false	false	false	false
"1"	true	false	true	false	false	true	false	false	false	false	false	false
"0"	false	true	false	true	false	false	true	false	false	false	false	false
"-1"	true	false	false	false	true	false	false	true	false	false	false	false
null	false	true	false	true	false	false	false	false	true	true	false	true
[]	false	true	false	false	false	false	false	false	true	true	false	false
"php"	true	false	false	false	false	false	false	false	false	false	true	false
" "	false	true	false	false	false	false	false	false	true	false	false	true

\* true prior to PHP 8.0.0.

Другая занимательная проблема присутствовала на широко известном в узких кругах сервисе Антикапча (сервис по разгадыванию капчи за деньги с API интерфейсом). Личный кабинет пользователя позволял совершать ряд операций, в т.ч. выводить неиспользованный баланс на WebMoney.

WebMoney нормально воспринимает сумму платежа в различных нотациях (например, со значением 1e1 или 0xFF), а сравнение подобных чисел, еще и на старых версиях PHP, еще и с учетом нюансов сравнения в языке PHP, еще и с порцией “качественного кода” приводило к самым неожиданным последствиям.

В шестнадцатеричной нотации сравнение текущего баланса с запрашиваемой на вывод суммой работало некорректно, что позволяло уводить баланс аккаунта в минус.

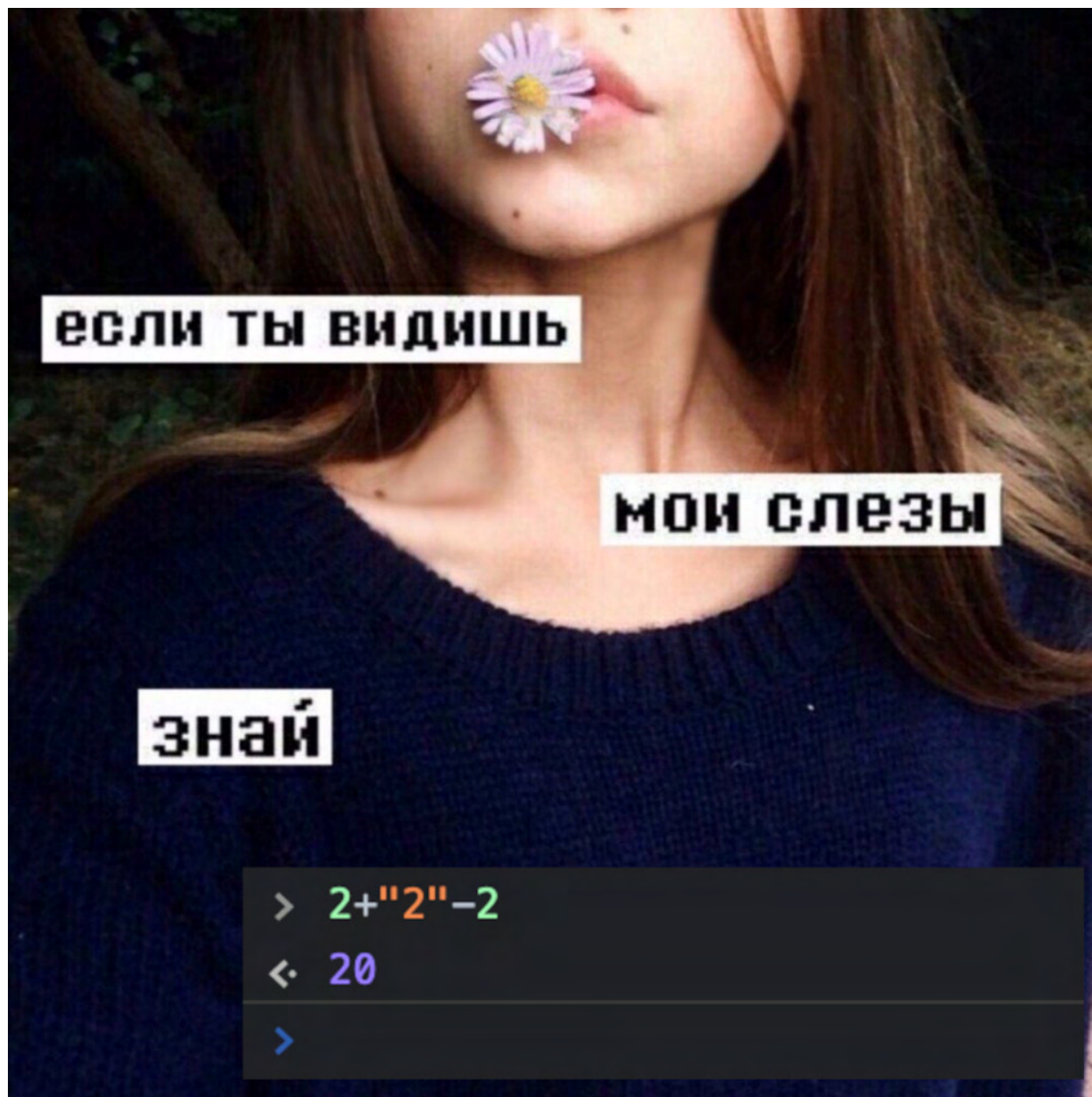
Пример возможной логики перевода денег.

```
$amount=$_POST['amount'];
$coins = preg_replace("/[^0-9]/", '', $amount);
if ($current_balance > $coins){
    perform_processing_and_withdraw_money_from_the_balance($amount);
}
```

Если на вход подать 1e9, имея на балансе 20 долларов, механизм проверки удостоверится, что 19>20, вырезав все кроме цифр, а процессинг обработает 1e9 как 1000000000.



Другая ошибка - это особенности приведения типов.



NodeJS - пример языка с динамической типизацией. Когда прибавляешь к числу строку, то произойдет конкатенация

$1 + "1" = "11"$

Но стоит из строки вычесть число, то уже строка приводится к числу

$"11" - 1 = 10$



Самый популярный формат обмена данными - JSON:

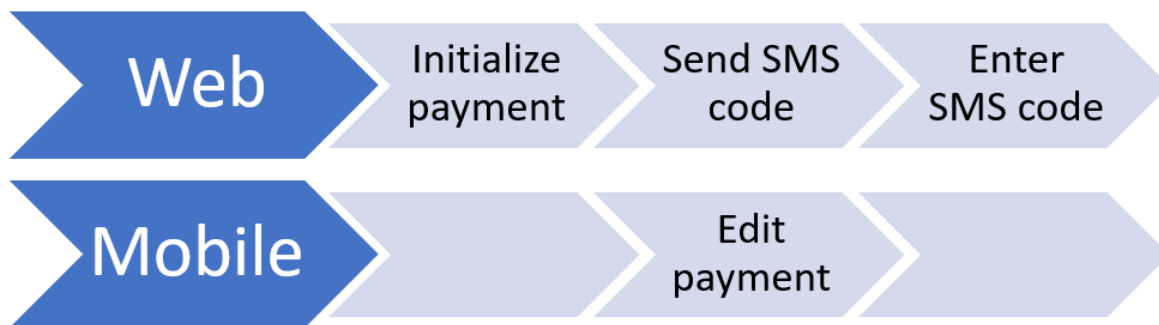
```
{"amount":100}
```

Справедливо, что JSON будет корректным: параметр amount с числом 100. Но и этот вариант будет корректным JSON'ом:

```
{"amount": "100"}
```

Только содержимое параметра amount будет строкой, но из-за особенностей обработки подобного запроса, возможно кто-то приплюсует к числу 1337 значение этого параметра, и получится 1337100, а не то, что задумывалось изначально)

## Ошибки бизнес-логики



В ДБО создается платеж, чтобы его подтвердить, необходимо ввести код из SMS. Платеж сохраняется как неисполненный и доступен для редактирования в мобильном приложении. Редактируем платеж, после этого в браузере вводим код подтверждения и итоговый перевод совершается с одной суммой, а со счета списывается другая.

Другой пример:

1. Вызываем механизм пополнения баланса (на балансе \$1000, пополняем на \$100).
2. Веб-приложение запоминает текущий баланс.
3. В это время тратим (отправляем на второй аккаунт) деньги.
4. После выполнения транзакции баланс окажется \$1100.

Отдельного упоминания заслуживает работа корзины на ресурсах, где используется несколько валют. Уязвимость которая была в магазине Xbox несколько лет назад (причем после исправления появлялась еще пару раз):

1. Кладёшь в корзину товар за минимальную цену в рублях.
2. Ищешь в магазине дорогие игры, цены на которые указаны в долларах.
3. Добавляешь их в корзину.
4. Магазин считает сумму позиций, но перерасчёт от доллара к рублю происходит в соотношении один к одному.

**Корзина** Продолжить покупки

Изображение	Название товара	Количество	Цена
	Rocket League® - Вулкан <small>Для всех возрастов</small> Цифровой продукт	1	400,00 P 50,00 P
	Minecraft for Windows 10 <small>16+</small> Цифровой продукт	1	26,99 P
	Call of Duty®: Infinite Warfare - Digital Deluxe Edition <small>18+</small> Цифровой продукт	1	52,49 P
	Полное издание Средиземье™: Тени войны™ <small>18+</small> Цифровой продукт	1	49,49 P

[Удалить](#) | [Сохранить для последующего использования](#)

**Сводка по заказу**

Товары (4):	178,97 P
Расчетный налог:	-
<b>Всего:</b>	<b>178,97 P</b>

[Оформить заказ](#)

Нужна помощь?  
Звоните **(8) 800 500 9242**  
Номер вашего заказа: 9489149131

Голоса в [vk.com](https://vk.com) генерировались с помощью SMS со счета с около-нулевым балансом. Отправляешь SMS, оператор связи не может забрать деньги (овердрафт отсутствует), а голоса пополняются.

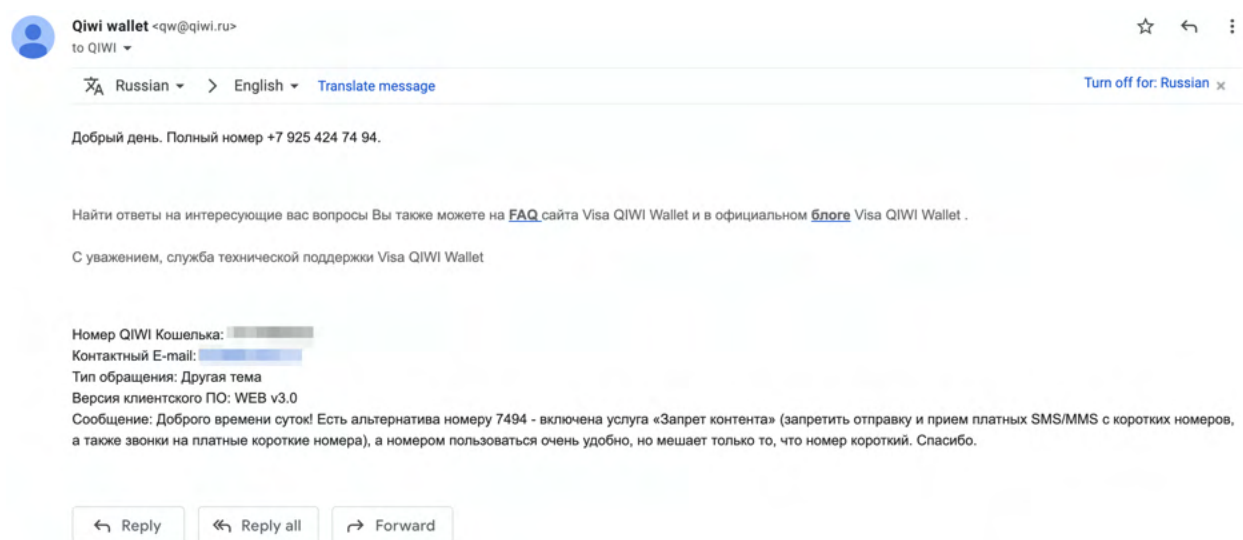
Другой вектор через SMS - это перевод со своего счета на чужой в платежной системе QiWI. Это делалось через отправку сообщения на специальный короткий

номер:

#### ПЕРЕВОДЫ

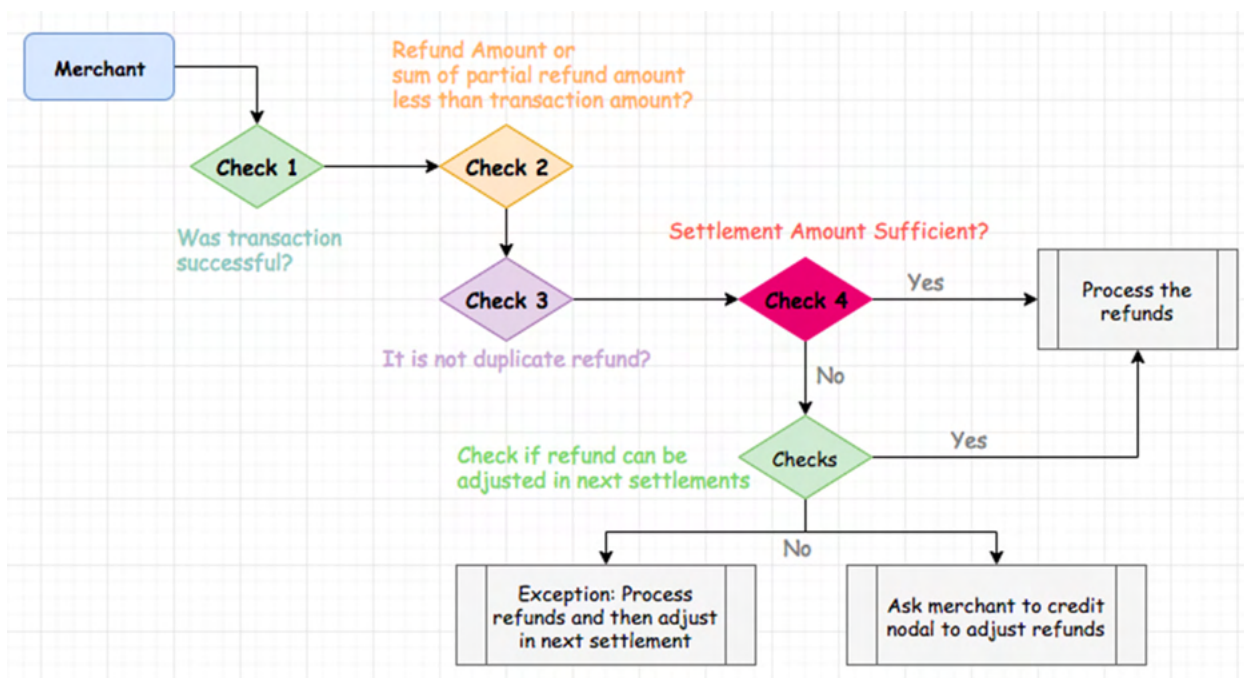
**Перевести деньги другому пользователю.** Отправьте на номер **7494** SMS с текстом **perevod** или **перевод**, через пробел укажите номер кошелька и сумму перевода. Например: perevod 9161234567 500. Вы получите SMS с одноразовым кодом – перешлите его в ответ.

Но дело в том, что короткий номер - это алиас настоящего телефонного номера, который участвует в SMS-шлюзе для интеграции с API. Применяем немного социальной инженерии:



Дальнейшие шаги — использовать сервисы по подмене номера, чтобы отправить туда SMS от аккаунта, на котором много денег. Способ так и не проверен, хотя в теории выглядит крайне забавно). Очевидцы говорят, что подобным образом можно было привязать карту через SMS, а дальше сливать деньги с карты.

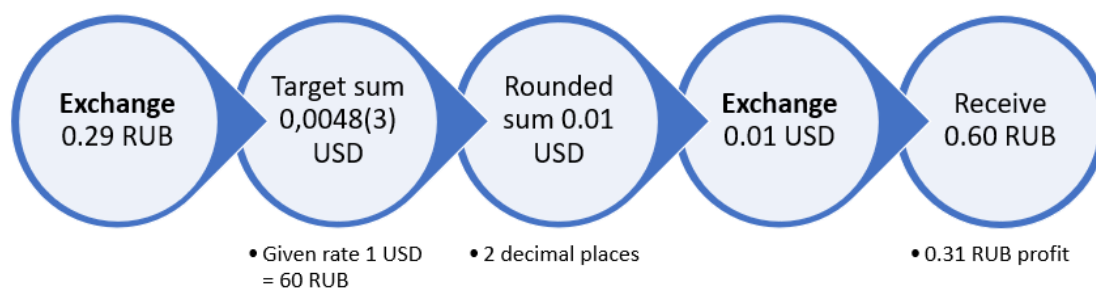
А теперь изучим операцию возврата средств (refund). Если рассмотреть каноничный процесс возврата, то станет очевидно, что на каждом этапе можно пропустить или некорректно реализовать все проверки, что приведет к финансовым потерям.



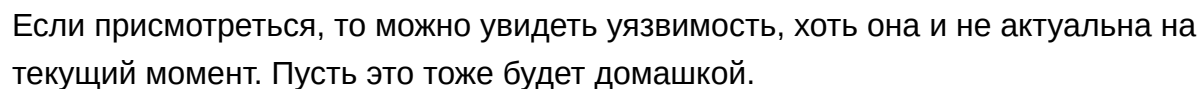
На практике встречались площадки, где возврат средств по транзакции происходил таким образом, что сумма возврата бралась от актуальной текущей стоимости товара, а не из информации о проведенной транзакции. Вместе с периодическими скидками, это приводило к понятным результатам. Ситуация редкая, но иногда встречается в той или иной форме.

## Ошибки округления, переполнения и числа с отрицательным знаком

Частая категория проблем - ошибки округления чисел. Распространенные проблемы с округлением могут выглядеть следующим образом:



- Проблему до сих пор можно встретить в крупных финансовых организациях (различных банках и биржах).



Понять, в чем уязвимость на скриншотах выше

С переполнениями и операциями с числами с отрицательным знаком при операциях также периодически можно столкнуться, даже в банках из топ 100. Перевод отрицательной суммы – тривиальный пример при работе с числами со знаком, и да, такое тоже **до сих пор встречается**.

Менее тривиальный пример про переполнения – подсчет суммы заказа при добавлении большого числа товаров в корзину.

Type	Storage size	Value range
char	1 byte	-128 to 127 or 0 to 255
unsigned char	1 byte	0 to 255
signed char	1 byte	-128 to 127
int	2 or 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647
unsigned int	2 or 4 bytes	0 to 65,535 or 0 to 4,294,967,295
short	2 bytes	-32,768 to 32,767
unsigned short	2 bytes	0 to 65,535
long	4 bytes	-2,147,483,648 to 2,147,483,647
unsigned long	4 bytes	0 to 4,294,967,295

Store credit:  
\$100.00  
Cart  

Name	Price	Quantity
Lightweight "1331" Leather Jacket	\$1337.00	- 17127 + Remove

  
Coupon:  
  
Apply  
  
Total: -\$20050873.96  
Place order

Еще один пример – это восприятие больших сумм при передаче между системами. В HTTP-запросе передаваемое число будет строкой, но вот обработка большого числа может отличаться, т.е. отправляется запрос на пополнение на больше чем  $\text{INT\_MAX}+2$ , на локальной системе число обрабатывается корректно, а в платежной системе получаем счет на оплату размером в 1\$.

Стоит учитывать, что тестируемая система может использовать не 32-разрядную переменную для хранения значения, а 64-разрядную.

Чтобы лучше понять, можно потыкать циферки в вконтакте. Раньше в [vk.com](https://vk.com) все числа были 32-разрядные. Чтобы получить id1 с помощью переполнения, необходимо было посчитать  $2^{32}+1$ .

Страница Дурова открывалась под <https://vk.com/id4294967297>

Но сейчас уже все переведено на  $\text{int64}$ , поэтому, чтобы получить единицу, необходимо посчитать  $2^{64}+1$

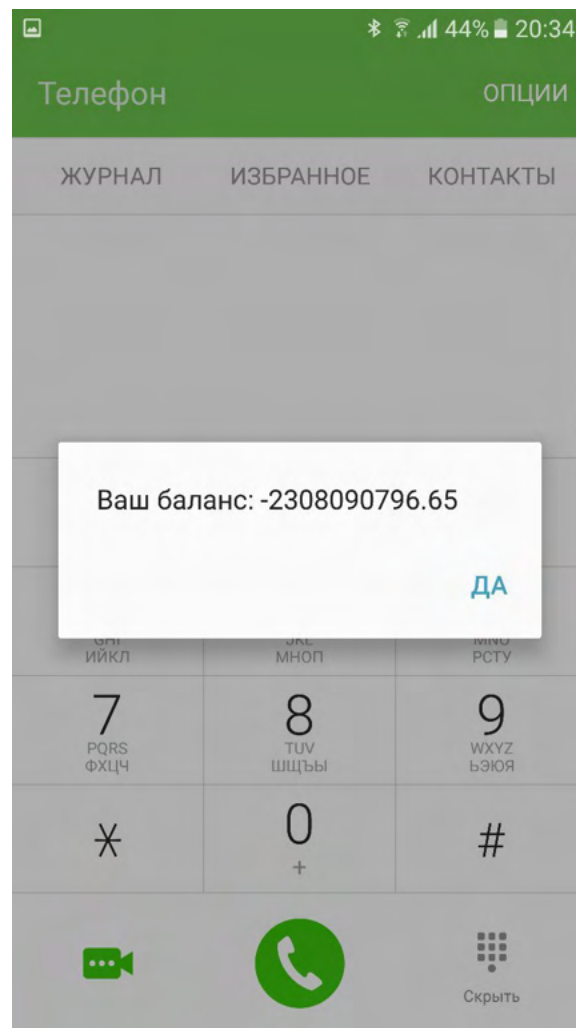
Это одни и те же страницы: <https://vk.com/id1> ==  
<https://vk.com/id18446744073709551617>

А теперь представь, что в веб-приложении операцию с id:100 может выполнить только администратор? А если это операция с  $2^{32}+100$ ?

Домашнее задание #3

У Дурова есть и другие "алиасы". Найди тот, что начинается на 3689

3.Ы. Иногда можно не рассчитывать с циферками, и уйти глубоко в минус, так и не достигнув плюса)

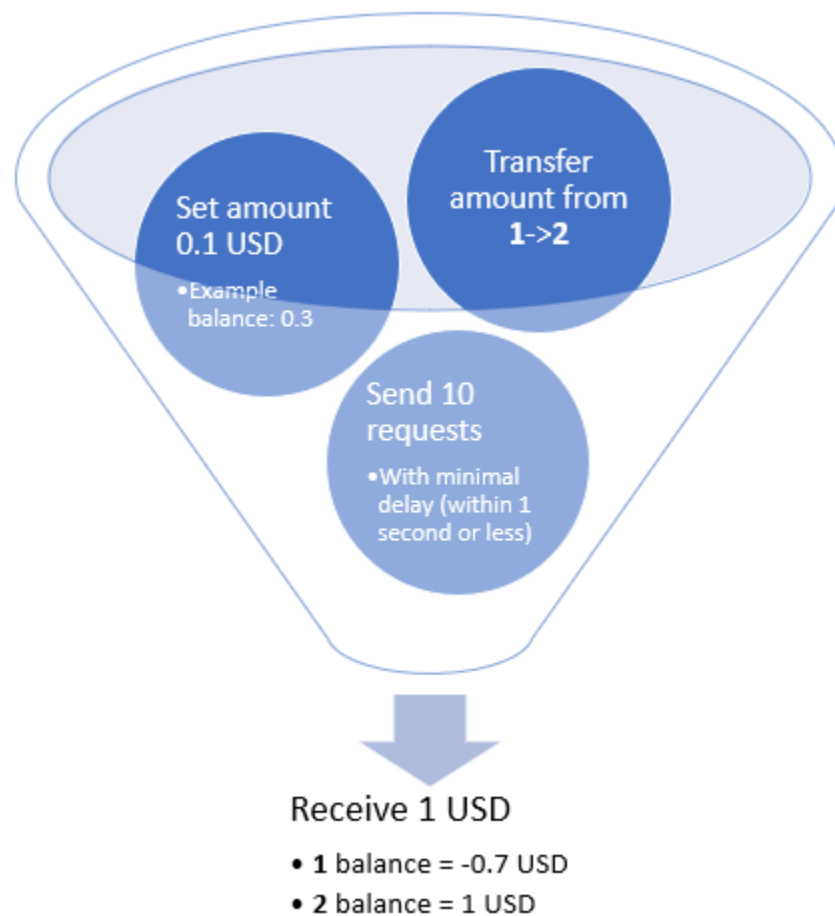


## Состояние гонки



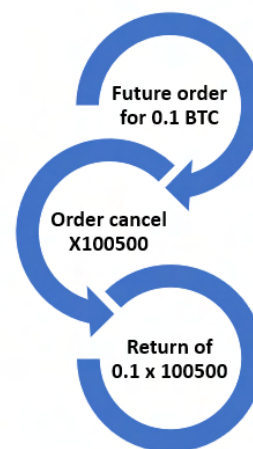
Перейдем к такой проблеме, как состояние гонки (англ. race condition). Согласно Wikipedia – это ошибка проектирования многопоточной системы или приложения, при которой работа системы или приложения зависит от того, в каком порядке выполняются части кода. Своё название ошибка получила от похожей ошибки проектирования электронных схем.

Условно каноничный пример:



1. Выполняем операцию на перевод средств в рамках баланса.
2. Совершаем ту же операцию N раз, где баланс должен закончиться при N-1 или больше, но отправляя запросы с минимальной задержкой (тут на помощь приходит HTTP-пайплайнинг, особенности с HTTP2 (все в рамках одной TCP-сессии) и т.п.).
3. Наблюдаем минус на балансе.

Небольшой пример, связанный с криптовалютной биржей.



Алгоритм эксплуатации был следующий:

- Создаем тейк-профит на 0,1 BTC, когда стоимость биткойна будет равна \$100,000
- Биржа изымает (блокирует) в аккаунте 0,1 BTC на будущий тейк-профит
- Удаляем тейк-профит отправляя 438695936458926734 запросов
- Биржа «возвращает» нам 0,1 x N BTC, где N, количество одновременно выполненных операций

Эта категория проблем не специфична для финансовых операций. Сюда же относятся проблемы типа TOCTOU, когда, например, приложение проверяет подпись на файле, далее некоторое окно и далее работа с содержимым файлом (а содержимое возможно подменить в рамках окна).

## Воруют деньги со счетов XSS.IS







Кстати, аналогичная проблема присутствовала на XSS.IS в системе перевода BTC между учетными записями.

Для тестирования можно использовать Burp Suite с плагином Turbo Intruder. А подробнее об этой категории проблем можно почитать в статье.








Для этого на депозит кладем 0.1337 BTC, отправляем множество запросов на перевод.

Row	Payload	Status	Words	Length ^	Time	Label	
0	198610%...	200	309	1157	321		
2	198610%...	200	308	1157	376		
3	198610%...	200	311	1157	361		
1	198610%...	200	449	1642	333		
4	198610%...	200	449	1642	352		
5	198610%...	200	447	1642	416		
6	198610%...	200	447	1642	470		
7	198610%...	200	450	1642	469		
8	198610%...	200	448	1642	464		
9	198610%...	200	448	1642	463		
10	198610%...	200	448	1642	730		
11	198610%...	200	447	1642	732		
12	198610%...	200	447	1642	702		

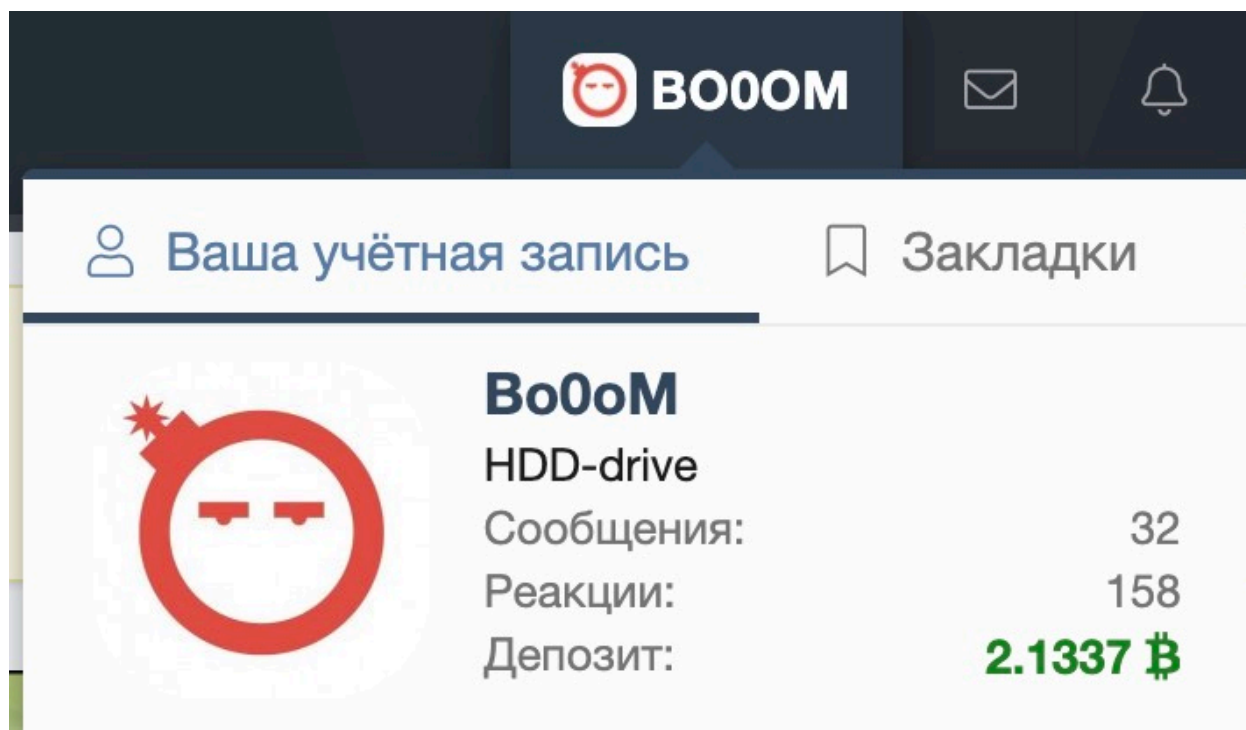
Видим, что функция отправки перевода выполнялась больше раз, чем было денег на балансе:

	Перевод пользователю Kaimi	0.1 ₿
	Перевод пользователю Kaimi	0.1 ₿
	Перевод пользователю Kaimi	0.1 ₿
	Перевод пользователю Kaimi	0.1 ₿
	Перевод пользователю Kaimi	0.1 ₿
<b>21 МАРТ 2020</b>		
	Пополнение	0.1337 ₿

Отправляем крипту обратно. Продолжаем гонять деньги туда-сюда под разными аккаунтами, генерируя деньги из воздуха:

	Перевод от пользователя Kaimi	0.7 ₿
	Перевод от пользователя Kaimi	0.7 ₿
	Перевод пользователю Kaimi	0.7 ₿
	Перевод пользователю Kaimi	0.7 ₿
	Перевод от пользователя Kaimi	0.5 ₿
	Перевод от пользователя Kaimi	0.5 ₿
	Перевод от пользователя Kaimi	0.5 ₿

Получаем деньги на счету. Только на самом деле такого депозита не было, поэтому выводить можно до тех пор, пока подключенный кошелек (со всеми депозитами пользователей) не опустеет.



Я думаю, есть и другие форумы, в которых возможны депозиты и автоматический вывод без ручного подтверждения.

Домашнее задание #4

Если это все время работало на [xss.is](https://xss.is), быть может, до сих пор работает и на других форумах?

## Резюме

Реализация безопасного приема платежей - это комплексная задача, которой должны заниматься опытные разработчики. Получившийся продукт необходимо всесторонне тестировать, иначе мы еще не один десяток лет будем наблюдать детские проблемы безопасности из начала нулевых, особенно при появлении новых классных способов платежей (привет, криптовалюта) и сопутствующих платежных систем. И это мы еще не упоминали атаки на генераторы псевдослучайных чисел, Padding Oracle, и множество других веселых штук, которые заслуживают отдельной статьи.

Kaimi & Bo0oM

Источник: <https://xss.is/threads/69067/>