作者：Kaimi & Bo0oM

电子支付系统在互联网上已经存在了很长时间，其中的一些漏洞已经有二十年之久。我们已经发现了一些关键的漏洞，使黑客能够盗窃资金并提高余额。今天我们将分析支付处理的典型实现逻辑和相关安全问题。

支付系统和典型API实现的概述

很少有人知道，但第一个（匿名的！）支付系统是DigiCash，它早在1989年就出现了，随后在1996年出现了一个更知名的（主要在持卡人中）系统E-gold。

但让我们回到现在，下面列出的是市场中主要的现代支付系统/电子支付服务，它们支持用户在网站上接受支付：

- PayPal
- WebMoney
- YooMoney (former Yandex.Money)
- Qiwi
- 支付宝
- etc.

以及几十个不太知名的系统，其名字你不可能熟悉，更不用说出现了数百个专门从事加密货币的新系统。

尽管表面上很简单，但就创建一个安全的软件实施而言，支付处理是一个复杂的过程，仍然给大型交易平台和新的电子支付系统带来问题，这些系统定期以"新的和方便的 "API和其他整合方式进入市场。一个典型的支付处理是什么样子的？首先，让我们看看PayPal描述的当前实施情况，即所谓的PayPal Express Checkout。

这种实现方式可以被认为是相对安全的，原因就在这里：

- 支付参数没有被明确传输，而是使用了一个token
- 支付系统的服务器不会自行将结果发送到某个URL，而是由你的网站来请求并处理响应。
- 总的来说，交互布局的实现方式是让潜在的开发者避免 "自伤"。

现在让我们来看看WebMoney提供的图示：

这个流程图没有任何意义。另外，该图没有反映出一些细微的差别，如请求签署。或者说，从支付系统接收技术支付信息的URL和用户被重定向到的URL（查看支付细节）应该是不同的。WebMoney使用的架构经常以这样或那样的形式出现在其他地方，通常出现在独立国家联合体创建的其他支付系统中。

典型问题

支付过程的过度复杂导致了财务损失。例如，10年前我曾发表过一篇关于全球收款服务系统与WebMoney整
合问题的说明，该系统允许在Steam
、Battle.net和其他一些平台上确认付款而不实际付款。

问题出在哪里？早些时候，我提到了商家方面应该接受支付信息的URL。根据文档，WebMoney有三个实体：

- 成功的URL -
  如果网络商户界面服务中的
  支付成功，买方的网络浏览器
  将被重定向到的URL（在商家的网站上）。该URL可以以 "http://"或 "https://"为前缀。
- 失败的URL - 如果网络商户界面服务中的付款由于某种原因没有完成，买方的互联网浏览器将被重定向到该URL（在商家的网站上）。该URL可以以 "http://"或 "https://"为前缀。
- 结果URL - 网络商户界面服务发送HTTP POST或SMTP通知的URL（在商户的网站上），该通知包含完整的细节。该URL可以以 "http://"、"https://"或 "mailto: "为前缀。

一些开发人员在阅读文档后会做什么？

1. 使用单一的URL，这样可以找出处理程序的地址（同时处理程序，包括结果URL，可以显示在WebMoney网站的支付表格中，但这并不总是发生，可能取决于设置）。
2. 不正确地对进入结果URL的请求实施签名验证。这使得客户可以替代支付细节。
3. 检查签名，但不检查发送到结果网址的金额。这允许你通过支付例如0.01美元获得100美元的物品。
4. 检查签名、金额，但不检查转移金额的格式。记得吗，我提到过通过客户的浏览器发送支付参数？WebMoney正确地处理了1e1或0xFF的值，但在旧版本的PHP上比较这种数字，考虑到PHP语言中比较的细微差别，会导致最意想不到的后果。
5. 这并不完全是支付系统的问题，但在商家的服务器上出现竞赛条件和相同的内部支付ID怎么办？你好，余额乘法。
6. ...

请求签名

它是如何工作的：

1. 当通过WebMoney支付时，根据支付系统的规格，用户被重定向到WebMoney网站，在那里他们可以看到支付的金额、账户号码和其他参数。
2. 在按下 "下一步
   "按钮并在系统中进行认证后，负责处理支付结果的URL的信息就可用了。
3. 用户可以向目标URL发出请求，根据WebMoney规范（好吧，几乎是），通知支付系统支付成功。
4. 获利！

Global Collect 支付处理系统遇到了几个问题：

1. 已知的统一的支付结果处理程序。
2. 缺乏签名验证（请求中也没有签名）。
3. 使用通过用户浏览器传输的数据作为支付信息的可信来源（尽管根据WebMoney规范，这可以通过来自WebMoney服务器的回调完成）。

所有这一切都允许进行虚假交易，购买任何使用全球收集处理的东西而不付钱。这个问题在被广泛利用了2周后才被消除。

另一个类似的问题，但更复杂一些，最近在Smart2Pay中被发现。

另一个与请求签名有关的问题是长度扩展攻击。

根据维基百科
介绍，这是对散列函数的一种攻击，在原始信息的末尾添加新的信息。在这种情况下，即使原始信息的内容仍然未知，也可以计算出新的哈希值。你可以在这里了解更多（https://web.archive.org/web/20130608035646/http://happybearsoftware.com/you-are-dangerously-bad-at-cryptography.html）。这个问题只在开发者决定实现他们 "酷 "的VK式请求签名时遇到过几次（顺便说一下，他们自己并没有发明一种算法）。下面是一个关于如何适当地生成签名以及如何 "自取灭亡 "的说明。

对于利用，你可以使用以下工具之一：

- https://github.com/bwall/HashPump
- https://github.com/iagox86/hash_extender

"结果URL "披露

在一个可以通过WooPay（通过短信）充值的网站上，支付系统用来通知网站是否成功入账的完整URL与参数（包括签名）一起被显示。

这个逻辑很简单，你需要想出一个办法来触发一个异常，使网页应用程序显示一个错误。

如果你选择通过短信支付，并随机输入一个无效的电话号码，你会得到：

重复 HTTP 请求几百次。禁止我们的客户端后，Web
应用程序抛出异常。它的文本包含秘密 URL。通过遵循它，我们可以完成付款。

支付属性

让我们继续讨论检查支付属性的问题。

与YooMoney（原Yandex.Money）整合的选项之一是汇款形式或其旧的实现方式。它可以通过存在对以下URL的HTTP请求来识别：

```
https://yoomoney.ru/eshop.xml
https://yoomoney.ru/quickpay/confirm.xml
```

当你发送请求时，你需要替换其中的付款金额：

支付成功的概率很高。然后，网站要么检查金额，要么不检查。由于支付元数据有用户ID和/或支付ID，这就足以购买东西。

一个小例子：

截图显示了Telegram语音助手机器人的订阅，但没有检查支付金额，允许你以任意的价格购买。将1990年改为19年，就能以真正的廉价获得它。这种漏洞相当普遍（例如，Telegram中有许多知名的机器人服务都存在同样的问题），包括在流行的国际资源上（一个老例子——购买Minecraft许可证）。即使在2022年，你仍然可以遇到这种情况。

另一个相关的例子，但与支付金额无关，而是与货币有关，出现在QIWI中。你可以通过向一个短号码发送短信来增加钱包的余额，货币在几个阶段（选择货币和金额，发送短信）通过客户端的浏览器传输，服务器相信客户端的数据。结果，100美元被记入账户，用于支付100卢布。

2022年的情况如何？

让我们来看看亚美尼亚银行的聊天记录https://t.me/Inecobank_forum/6333 ：

> 你好! 有什么办法可以将卢布账户的钱立即转入银行卡的德拉姆账户（绕过目前银行的德拉姆账户），从而不违反俄罗斯联邦的货币立法？

> 一般来说，在我被告知这是不可能的之前，我使用其详细信息从我的活期账户向银行卡账户转了10万卢布，以德拉姆计算。100,000德拉姆已经入账。10万卢布被扣除了。都是在INECO。很明显，没有自动验证支付货币的功能。现在我正在与支持部门打交道。Moral--不要这样做。

这意味着问题仍然存在。

格式和类型比较

在Anticaptcha服务上发现了一个有趣的问题，这个服务在某些圈子里很有名（一个用API接口解决验证码赚钱的服务）。用户的个人账户允许执行一些操作，包括将未使用的余额提取到WebMoney。WebMoney很正常地处理不同格式的支付金额（如1e1或0xFF），但比较这些数字，特别是在旧版本的PHP中，考虑到PHP中比较的细微差别，拥有 "最完美的代码"，导致了最意想不到的后果。在十六进制的符号中，当前余额与要求提款的金额的比较不能正常工作，这使得账户余额变成负数。

一个可能导致这种情况的PHP代码片段的例子：

如果你输入金额1e9，余额为20美元，验证逻辑将确保19<20，删除除数字以外的所有内容，但支付系统将把1e9视为1000000000。

另一个错误是类型化的特殊性

NodeJS
是另一个具有动态类型的语言的例
子。当把一个字符串加到一个数字上时，它会把1+"1"="11
"连接起来。然而，如果我们从字符串中减去一个数字，字符串就会转换为数字
"11"-1=10。

最流行的数据交换格式是JSON：

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

很明显，这个 JSON 是正确的：有一个值为 100 的 amount
参数。但这也是一个有效的 JSON：

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

这里，金额参数的类型是一个字符串。根据JSON处理的算法（顺便说一下，有趣的相对相关的文章：https://bishopfox.com/blog/json-interoperability-vulnerabilities），有人可能会把这个参数的值加到数字1337上，结果会是1337100，而不是原来的意思。

商业逻辑的缺陷

在远程银行
中启动了一项付款，为了确认它，你需要输入短信中的代码。付款被保存为未完成，可在远程银行手机应用程序中进行编辑。我们编辑付款，然后在浏览器中输入确认码，最后转账的金额是一个，但账户被扣了另一个。

另一个例子：

1. 打开余额充值界面（余额为1000美元，充值金额为100美元）。
2. 网络应用会记住你当前的余额。
3. 同时，我们花钱（发送第二个账户）。
4. 完成交易后，余额将是1100美元。

同样值得特别一提的是，在支持多种货币的资源上的购物车逻辑。几年前在Xbox地

区商店发现的一个漏洞（修复后又重新出现了几次）。

1. 你以最低的卢布价格将产品添加到购物车中。
2. 你在商店里寻找昂贵的游戏，这些游戏的价格是以美元列出的。
3. 把它们添加到你的购物车。
4. 商店计算商品的总成本，但美元价格的商品会以一比一的比例重新计算成卢布。

上面的截图显示，购物车中的游戏费用是以卢布表示的，但实际金额暗示应该是美元（或者按适当汇率转换后应该完全不同）。

Vkontakte
上的投票是在余额接近零的情况下通过发送付费短信添加的。你发送短信，电信运营商不能拿钱（没有透支），但票数还是被补充了。

另一个通过短信的载体是在QIWI支付系统中从你的账户转到别人的账户。这是通过向一个专门的短号码发送消息来完成的：

> 转账给另一个用户。发送短信至7494，短信内容为 "perevod "或 "转账"，输入钱包号码和转账金额，用空格隔开。例如：perevod 9161234567 500。你会收到一条带有一次性代码的短信--把它回复回去。

该短号码实际上是一个真实电话号码的别名，该号码被短信网关用于与API的整合。对支持服务应用一点社会工程：

> 下午好。完整的号码是+7 925 424 74 94。
>
> 请求类型：其他主题。
>
> 客户端软件版本：WEB v3.0。
>
> 消息：下午好! 有没有7494号的替代号码--"内容阻断 "服务已启用（禁止

> 从短号码发送和接收付费短信/彩信，也禁止拨打付费短号码），使用该号码非常方便，但由于该号码很短，所以无法使用。谢谢你。

下一步是使用欺骗服务（欺骗来电显示），从一个有很多钱的账户的号码发送短信。这种方法我从来没有测试过，尽管在理论上它看起来非常有希望。一些专家说，有可能通过短信链接信用卡（以类似的方式），然后从卡上抽走钱。

现在我们来研究一下退款操作。如果你考虑典型的退款过程，就会发现在每个阶段都有可能跳过或不正确地实施一些检查，这将导致经济损失。

在实践中，有些网站的退款金额等于产品的当前价格。相反，他们应该使用相关交易记录中的实际支付金额。再加上定期折扣，这导致了明显的结果。这种情况很罕见，但有时会以这样或那样的形式出现。

四舍五入、整数溢出和负数

一类经常出现的问题是四舍五入错误。常见的四舍五入问题可能是这样的：

1. 一个用户将0.29卢布兑换成美元。
2. 如果一美元的价值是60卢布，0.29卢布的金额对应于0.004833333333美元。
3. 这个数额将被四舍五入到小数点后两位，即0.01美元（1美分）。
4. 然后，用户将0.01美元换回卢布，得到0.60卢布。
5. 因此，用户 "赢得 "0.31卢布。

这个问题在大型金融机构（各种银行和交易所）仍然可以找到。

如果你仔细观察，你可以看到这个漏洞，尽管它已经被修复。负数交易的溢出和漏洞可以定期遇到，即使是前100名的银行也是如此。在处理带符号的数字时，带负数的交易是一个微不足道的错误，是的，它仍然发生。

一个不那么微不足道的溢出例子是在向购物车添加大量物品时计算订单金额。

另一个例子是在系统间传输时，大数字的兼容性问题。在HTTP-request中，传输的数字将是一个字符串，但对大数字的处理可能是不同的，即发送一个超过INT_MAX+2的充值请求，在本地系统中，数字的处理是正确的，但支付系统收到的是1美元的支付发票。

请记住，目标系统可能没有使用32位变量来存储数值，而是使用64位变量。

为了得到更好的理解，你可以在Vkontakte上玩玩数字。VKontakte早期使用32位整数。为了通过溢出的id达到id1页面，必须通过2^32+1。

Pavel Durov的页面可以通过以下网址打开：https://vk.com/id4294967297。但现在一切都被转换为64位整数，所以要得到id1，需要传递2^64+1。

这些是相同的页面。

https://vk.com/id1 == https://vk.com/id18446744073709551617

现在让我们考虑一下，在一些网络应用中，一个id=100的操作只能由管理员执行。那么如果这是一个2^32+100的操作呢？

顺便说一下，有时你可能会计算错误，出现严重的负数，永远无法获得利润。

竞赛条件

让我们继续讨论竞赛条件问题。根据维基百科介绍，竞赛条件或竞赛危险是指电子、软件或其他系统的状况，其中系统的实质性行为取决于其他不可控事件的顺序或时间。当一个或多个可能的行为是不可取的时候，它就成为一个错误。

一个传统的典型例子：

1. 我们执行一项交易，从可用余额中转移资金。
2. 我们执行同样的操作N次，让我们考虑我们应该在第(N-1)个请求或更早的

时候用完余额。以最小的延迟发送请求可能会利用竞赛条件并克服这一限制（这里HTTP-
pipelining，HTTP2功能（在一个TCP会话中的多个请求）等来拯救）。
3. 我们会得到一个负的平衡，这在正常情况下是不允许的。

一个与加密货币交易所有关的小例子

操作算法如下：

1. 当比特币的价格为100,000美元时，我们创建一个0,1 BTC的获利。
2. 交易所从账户余额中提取（锁定）0.1 BTC。
3. 我们通过发送438695936458926734个请求来删除该获利。
4. 交易所 "返回 "0.1 × N BTC，其中N是同时操作的数量。

这类问题并不是专门针对金融交易的。这也包括像TOCTOU这样的问题，例如，当应用程序检查文件签名，然后过了一段时间，然后应用程序读取文件的内容（可以通过该时间来替代）。

其中一个问题是在xss.上出现的，是在账户之间转移BTC的系统。

你可以使用Burp Suite与Turbo Intruder插件进行测试。而你可以在这篇文章中阅读更多关于这类问题的内容：https://lab.wallarm.com/race-condition-in-web-applications/。

因此，我们存入0.1337个BTC，并发送大量的转账请求。

我们可以看到，转账完成的次数超过了余额有限的情况：

我们将加密货币发回。我们不断在不同账户之间来回转移资金，凭空产生资金：

最后，我们得到了更多的余额（2.1337 BTC）。然而，实际上并没有这样的存款，所以你可以提取与连接的钱包一样多的钱（所有用户存款）。

我想还有其他论坛的存款和取款是可以不经人工确认的。

总结

实施安全支付处理是一项复杂的任务，应该由有经验的开发人员来处理。由此产生的产品需要进行全面的测试，否则我们将在未来的几十年里观察到九十年代初的幼稚的安全问题，特别是当新的很酷的支付方式（你好，加密货币）和相关支付系统出现时。而我们甚至还没有提到对伪随机数生成器、Padding Oracle的攻击，以及其他许多值得单独写一篇文章的有趣东西。