(from @jaosef) there are several sources of potential bugs that an attack could exploit:

The following can cause an exploit

1. Kernel Circuit Read Request
2. Kernel Circuit Write Request
3. Nulifier bug in rollup circuit
4. Insecure Nulifier scheme used by app in Note Implementation
5. Aztec.nr bug
6. App Circuit Logic bug

How does an observer distinguish between these sources when diagnosing the source of an attack?

Let's start by assessing the properties we desire. Naturally it will be possible to diagnose the source of the bug by conducting a code review and finding

the bug, but this will likely be a long time (>1 week) after the attack has occurred. Are there heuristics we can apply to reduce this time gap?

Protocol bugs can lead to the following:

1. validate a private note exists when it does not exist
2. validate a private note does not exist when it does
3. enable encrypted notes to be decrypted + leak secrets (bug in encryption algorithm)
4. change the time-ordering of function calls in a transaction (e.g. instead of function A calls function B, tx executes B before A)
5. misrepresent the identity of a user (msg.sender

)

The above is what currently comes to mind, please add to this if you can think of more!

I want to focus on points 2 and 3 as these are where subtle bugs can hide. As protocol designers we have a responsibility to ensure none of these cases can ever be triggered, but we are messing up in a truly exceptional manner if points 3, 4, 5 can be triggered.

Double-spends from the protocol vs double-spends from the application

One obvious way of determining protocol-vs-application is that a protocol vulnerability will lead to multiple apps being exploitable, so an attack may simultaneously attack multiple contracts.

However, the network will likely support some kind of pause functionality (see upgrade proposals), which may incentivise an attacker to drip attacks into the network 1 contract at a time to prevent an emergency pause from triggering.

My initial take is that, without some context behind how any individual app functions, it's not possible to create a mechanism which distinguishes whether the site of an attack lies at the protocol or the app level.

However I think we can come up with a checklist to follow when diagnosing attacks. For the time being I think this is the best that we can do? This is a very incomplete list so far, would appreciate thoughts on fleshing it out!

1. Examine the UTXO objects created by the app. How are they encrypted? Are validity checks correct applies?
2. Examine the private->public function calls that are conditioned on the plaintext of UTXO objects. Are validity checks+logic checks consistent with intended functionality?
3. Is the app using a custom nullifier scheme instead of the default one as defined by Noir's Note implementation? If so, is the scheme sound?