TL;DR:

Time server attacks on Eth2 have been considered in the past. It was so far considered that the attack vector would only be able to knock validators offline temporarily, until correct clock synchronization is restored by the node operator. However, a bigger danger lurks if nodes can be tricked into signing an attestation for a far future epoch: due to surround vote slashing, they would not be able to sign any attestation until the time of the future attestation target. For the purposes of consensus, this knocks them out almost permanently, and will be costly for the staker due to inactivity leaks.

Attack vector

Attestations in Eth2 are a signature on a source and a target checkpoint, each consisting of an epoch number and a block root. The safety of the Casper FFG consensus is ensured by the “no surround vote” rule: No validator should ever sign two attestations where the attestation1.source < attestation2.source

and attestation2.target < attestation1.source

. [1]

We can craft an NTP attack as follows: The first step is to set the attacked validators time to the far future, by manipulating the time of the NTP server to be in the future, for example 15-20 days (it needs to be less than the time of the inactivity leak drawing balances to zero, otherwise the validator will not sign attestations).

Then we need to trick the corresponding node to think that it is synced with the chain so that it will sign attestations. This should always be possible if the attacker controls some of the peers, which is so easy to achieve that it can always be assumed, by sending some attestations and blocks from the future time on the P2P channels.

Once the validator has signed an attestation for the future epoch, the attacker can store this attestation and knows that this validator cannot sign any attestation for the current chain until that future epoch kicks in, otherwise it can be slashed immediately. The Validator Client node of all current Eth2 implementations will actually prevent it from ever signing a conflicting attestation, thus the validator will be effectively offline.

Result

An attacker can likely knock out all validators it can get control of via an NTP (or similar, e.g. roughtime) attack vector. This attack is worse than previous attacks described using this vector [2], because the result is more permanent: It is likely that an NTP attack will be detected within minutes and almost all nodes with competent operators should be back to normal withing hours. However in this case that won’t help as the damage is already done – and can cause very large financial damage to validators, e.g. if enough validators are knocked out to lead to a quadratic inactivity leak.

Mitigation

Add VC no-slash rule

We can add a no-slash rule to Validator Clients that requires them to evaluate the current time before signing anything and not sign any attestations for the future. This only helps if the VC runs on a different node than the BN, and is not affected by the attack. However, it would be a strict improvement in the case of Secret Shared Validators, as it does not allow the leader node to propose a far future attestation and block the validator.

Add more robustness to time synchronization

This issue highlights that NTP time synchronization is a more serious attack vector than previously thought, and care has to be taken to avoid this.

Small clock skews are annoying but do not cause serious harm. It is thus preferable to only use NTP time updates if they are within a small window of current RTC time (seconds), and otherwise reject them.

This leaves the boot process as an attack window. Note that major power outages do happen and this is still a potentially serious issue that can affect many validators at once. I think it might be a good practice for validator clients to refuse launching (except with an override flag) if the slashing protection database indicates that nothing has been signed for several hours or days, indicating the possibility of an NTP attack.

(Thanks to @alonmuroch for a discussion that led to the discovery of this attack)

[1] https://arxiv.org/abs/1710.09437

[2] Time attacks and security models