Simple attack

Hackers have a simple and rational incentive to attack validators: with the validator's private key, he can generate slashable attestations and claim the corresponding "whistleblower" reward.

The hacker does not need to claim the reward immediately. So if he finds a zero-day attack on an eth2 client, he can quietly exploit it on all the validators he can find on the network, collecting all the private keys before claiming all rewards at the same time.

As the victim, if you find out you have been hacked, the optimal strategy is to slash yourself as fast as possible and claim the whistleblower reward --the staked funds are lost anyway.

The hacker's whistleblower reward is limited (~0.05 ETH, ~€10), but already interesting if you can hack a few thousand validators.

Blackmailing

As the cost for the victim is high (from 1 to 32 ETH). The attacker can use a blackmailing smart contract to increase its profit. Rather than slash the victim, the attacker extorts the victim. The deal is secured with the smart contract so the attacker is paid only if the victim is not slashed.

Once the attacker controls the victim's private key:

1. He proves, off-chain, he has the private key by signing a random message.
2. He asks the victim to send 50% of the slashable funds to a specific blackmailing smart contract. On this smart contract one can:
3. Get the funds back to the sender by proving that the corresponding validator has been slashed.
4. Transfer the funds to the attacker by proving that the corresponding validator has exited or that more than a year has passed.
5. The victim has to choose whether to pay or not.

Eth2's slashing mechanism is non trivial. The minimum slashed amount is 1 ETH, but increases if anyone else is slashed during the next 18 days. We suppose that the attack is large enough to ensure that everybody is slashed to the maximum (i.e. 32 ETH).

As of today, for a validator slashable funds are 32 ETH, the whistleblower reward 0.05 ETH (0.0546875 exactly, (32/512)* (7/8)). If the victim exits without being slashed he will have a revenue of 32 ETH (his own funds given back). The table actions/revenue is:

Victim

Attacker

Victim pays and exits peacefully with his stake

16

16

Victim does not pay and get slashed by the attacker

0

0.0546875

Victim does not pay and slashes himself

0.0546875

0

Victim pays but gets slashed by the attacker

0

0.0546875

Victim pays and slashes himself later

0.0546875

0

We see the victim's optimal strategy is to pay. This maximizes the attacker's profit as well.

Once the victim has paid, the optimal strategy for the attacker is to wait for the funds to be transferred. This maximizes the victim's profit as well.

Moreover, this attack scales: as in the initial scenario, if a zero-day security issue is identified, the attacker can gather all the private keys for all validators found on the network, then and only then, contact the victims.

Of course the hacker can increase the price up to ~31.9 ETH and it is theoretically still optimal for the victim to pay. In real life it's likely too much.

Blackmailing in the dark

The attacker can as well extend his attack by not proving he knows the private keys for all potential victims. Let's say (1) that the attacker took control of the private keys of 25% of the potential victims, and (2) that the potential victims cannot determine if they have been actually hacked or not.

He would then be able to blackmail his victims in the following way: "I know your private key. Give me 16 eth or I'll impersonate/slash you. You may ask for a proof of possession but that will cost you 3 extra eth."

If the actors are rational, this is strictly better than the previous approach, as the hacker will get paid for all the private keys hacked, and some of the ones he hasn't actually hacked.

Ultimately, the attacker does not even have to hack anything but can simply do a public relation stunt by pretending he took control of private keys and slashing validator keys he created himself to create some credibility.

Having devices specialized for signing helps tremendously against this attack, but does not fully break it: if the attacker can't access the private key during the attack, he can still generate slashable attestations, and store them for later use. It is still an improvement as this unexpected activity could be detected. A signing device that can detect and refuse to sign conflicting messages would work.

(thanks to @AlexandreBelling, @benjaminion, @OlivierBBB for the review/comments)