DoS on a Bitcoin Lightning Network channel.

Willem Rens (UvA MSc SNE student)

Supervisors: Maarten Everts (TNO) & Oskar van Deventer (TNO)

August 2, 2018

Abstract

The purpose of this research is to identify whether it is possible to steal bitcoin on the Lightning Network by means of a successful DoS attack. First of all an attack to do so is presented in theory. In addition this attack is demonstrated in practice using Lightning Labs Lightning Network daemon on Bitcoin's testnet. Finally the results show that it is possible to steal bitcoin on the Lightning Network in case of a prolonged successful DoS attack.

Contents

1 Introduction

Bitcoin's Lightning Network[1] has been proposed as a layer on top of its blockchain to facilitate higher transaction throughput. According to the Lightning Network paper authors it will help scale to "billions of transactions per day with the computational power available on a modern desktop computer today. (2016)". The network is formed by bidirectional payment channels, in which valid Bitcoin transactions may be exchanged. However, they do not have to be broadcast to the Bitcoin network, which prevents blockchain bloat.

This research investigates a vulnerability in its design, by which bitcoin in a channel belonging to a counterparty can be claimed in case of a successful denial-of-service(DoS) attack. It will provide 1) an exploration of this attack in theory; 2) a simulated attack using Bitcoin's testnet3 and the Lightning Network daemon[1] to test its practical feasibility. Furthermore, a discussion on watchtowers, Blockstream's Eltoo proposal and a formula that estimates the upper limit costs for the attack to be profitable is presented.

The next part of this paper will provide background information, which includes an introduction about the Lightning Network, the origin of suspecting this vulnerability and related work. After this, the research questions are presented and the approach to answer them is described afterwards. Results can be found in the succeeding sections, followed up by a discussion and a conclusion. Finally, suggestions for future work are given.

2 Background information

This section starts with a conceptual overview of the Lightning Network. A technical view about how channel states are managed is provided in section 5. It is expected of the reader to have a good knowledge of Bitcoin.

2.1 The Bitcoin Lightning Network

Bitcoin's Lightning Network attempts to provide a scalable, trustless and low-latency payment network using Bitcoin's blockchain as its arbitration layer[2][3]. Fundamentally, it works by making use of bidirectional payment channels and Bitcoin's scripting system. To create such a channel, two parties create a funding transaction by sending an amount of bitcoin to a multisignature output. Updates to balances in the channel are made by creating a new commitment transaction that spends from the initially created funding transaction. Since the funding transaction is a multisignature output, it requires signatures from both (or more) participants to spend from this output. Note that these transactions are all real Bitcoin transactions and may be broadcast to the blockchain at any time. However, by invalidating old channel states by means of a penalty system, users are motivated to only broadcast the most recent state. Section 5.1 will describe this process in more detail.

[1] https://github.com/lightningnetwork/lnd

An innovation of the Lightning Network is to form a network as such that payments can be routed through multiple channels (nodes), which are not directly connected. This is enabled by making use of a transaction called a Hashed Timelock Contract (HTLC)[8].

2.2 DoS issue on the Lightning Network

Peter Todd, a long term Bitcoin developer, has raised concerns about the possibility of the Lightning Network being vulnerable to DoS attacks[4]. Furthermore, the authors of the Lightning Network paper state, in section 3.3.4 of their paper that: "One should periodically monitor the blockchain to see if ones counterparty has broadcast an invalidated Commitment Transaction". Moreover, in section 9.5 they claim that: "If one does not broadcast a transaction at the correct time, the counterparty may steal funds.". This leads one to believe that if a counterparty is hit by a successful DoS attack, funds can be stolen.

2.3 Related work

McCorry et al. dissect different payment channels and regarding Lightning Network channels they state: "The revocation mechanism requires both parties to check the Blockchain periodically to detect if a previously revoked channel state has been submitted"[7].

BitPico has claimed to have run a DDoS attack against the Lightning Network on Bitcoin's mainnet: "Operating out of 8 countries running 22 attack vectors in-parallel from 384 endpoints."[5]. Resulting in sending around 20% of nodes offline[6]. Their research has shown it is possible to send down a significant amount of lightning nodes.

Blockstream has proposed a new mechanism for transitioning between channel states, which they call eltoo[9]. An advantage is that old states do not have to be stored. Since this new mechanism might be added to the Lightning Network, in section 7 it will be briefly discussed in relation to this paper's research.

3 Research questions

1. Can bitcoin in a Lightning Network channel be stolen by a successful DoS attack?

2. Is it possible to carry out this attack in version 0.4.2 of the Lightning Labs Lightning Network daemon?

The scope of this research is limited to a single payment channel between two participants. Multi-hop payments, which make use of HTLC's, are not considered in this paper.

Note that a partial DoS may also be classified as a successful DoS attack. This paper, however, uses the term successful DoS attack in a way that means a complete DoS of its target.

4 Approach

To answer the first research question the Lightning Network paper is explored on the design of preventing old channel states to be accepted into the blockchain. This is done from the perspective of a single payment channel between two participants: Alice and Bob. Next, the attack on this design is presented in the perspective of Alice and Mallory, in which Mallory is the malicious actor who attempts to claim funds that are not his.

The second research question is answered by demonstrating this attack in practice using Bitcoin core v0.16.0[2] on Bitcoin's testnet3, together with version v0.4.2-beta of the Lighting Labs Lightning Network daemon[3](LND). See Figure 1 for an illustration of this set-up. The testnet uses a different blockchain than the real Bitcoin; the coins can be claimed for free from a so called faucet [4]. A channel is set-up between Alice and Mallory, in which Mallory deposits 0.01206694 BTC. Note that this amount has not been chosen arbitrarily, it is the highest channel value to receive the smallest timelock possible. Then Mallory will pay Alice several times using this channel, resulting in Mallory having a balance of 0.00012512 BTC. After that, Mallory will start a successful DoS attack on Alice, which is simulated by disabling Alice's networking interface. Now Mallory will try to get her 0.01206694 BTC (minus transaction fees) bitcoin back by trying to get an outdated state of the channel accepted into Bitcoin's testnet3 blockchain.

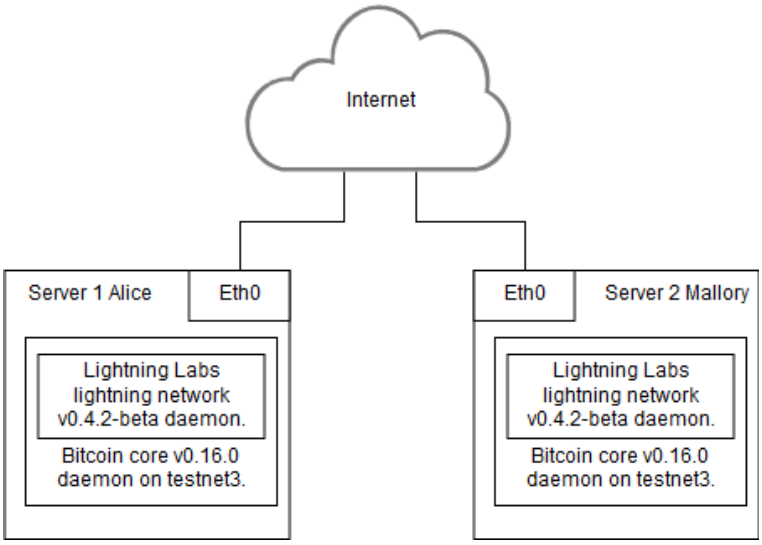

Figure 1: The server set-up during the experiment(s). Server 1's Eth0 interface will be disabled to simulate a successful DoS attack.

[2]https://bitcoin.org/en/release/v0.16.0
[3]https://github.com/lightningnetwork/lnd/releases/tag/v0.4.2-beta
[4]https://testnet.manu.backend.hamburg/faucet

5 Attack in theory

This section explores the attack in theory, starting with a dissection on the mechanism proposed by the Lightning Network paper authors to invalidate old channel states.

5.1 Mechanism of invalidating old states proposed in the Lightning Network paper

A channel state is expressed as a valid Bitcoin transaction that is signed by both participants, which is called the commitment transaction. It pays out the respective current balances to each party. When both parties want to update the channel state, i.e. their respective balances, they create a new commitment transaction. To prevent an old commitment transaction from being broadcast, an attempt is made to give all funds to the other actor as a penalty when done so. It is made possible by making one of the outputs a Revocable Sequence Maturity Contract (RSMC). This is combined with creating two different commitment transactions, one for each actor, as such that each actor only has access to a completely signed commitment transaction in which their own funds are placed in a RSMC output. This is illustrated in Figure 2.

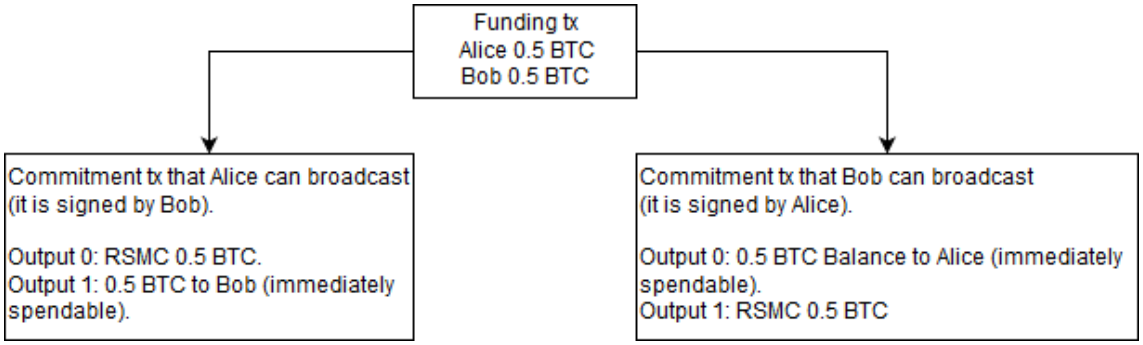

Figure 2: The channel state is expressed as two commitment transactions, one that Alice may broadcast and one that Bob may broadcast. One of the outputs is a RSMC in an attempt to invalidate old channel states.

The RSMC delivers an output that is spendable in two ways. If Alice has broadcast the commitment transaction, this output may be spent by Bob using a Breach Remedy Transaction(BRT), which is possible if he knows the revocation private key. Or it is spendable by Alice after a relative time-lock has finished, enforced by OP_CSV. This relative time-lock works in the sense of block height in relation to the block height the commitment transaction was confirmed at. The RSMC contract can be seen in the listing below[10]:

```
OP_IF
    #1 Breach Remedy Transaction(spendable by Bob if revocation private key
```

```
    # is known)
    <revocationpubkey>
OP_ELSE
    #2 Time locked transaction(spendable by Alice if time-lock has finished)
    `to_self_delay`
    OP_CSV
    OP_DROP
    <local_delayedpubkey>
OP_ENDIF
OP_CHECKSIG
```

Thus, to attempt to invalidate an old state, actors exchange a revocation private key needed to create the BRT. The counterparty of the broadcaster is now able to spent the balance in the RSMC output using the BRT, while the broadcaster has to wait the relative time-lock duration. Because Alice and Bob each have their own version of the commitment transaction, they can only broadcast the BRT if the **other** actor has broadcast an old commitment transaction. It follows that, broadcasting an old commitment transaction gives the other actor an opportunity to claim all the funds in the channel.

5.2 The Attack

The attack is presented as a step by step list:

- Mallory opens a channel with Alice. Mallory balance: 0.1 BTC — Alice balance 0 BTC
- Mallory saves this state, i.e. the commitment transaction signed by Alice and the OP_CSV locked output sweep transaction.
- Mallory pays Alice in one or more transactions in return for services or goods. Mallory balance 0 BTC — Alice 0.1 BTC
- Mallory starts a DoS on Alice and broadcasts the old commitment transaction (Mallory balance: 0.1 BTC — Alice balance 0 BTC) and carries out a DoS attack on Alice for the CSV lock duration + time until output sweep transaction has one confirmation

In essence the attack is to broadcast an old commitment transaction, which has a balance that is favorable to Mallory compared to the latest state. To prevent Alice from broadcasting the BRT, a DoS attack is used for the OP_CSV lock duration + time until the output sweep transaction has confirmed. Figure 3 illustrates this in generic terms. Furthermore, Figure 4 shows the different kind of channel closures, having marked in red the steps an attacker takes.

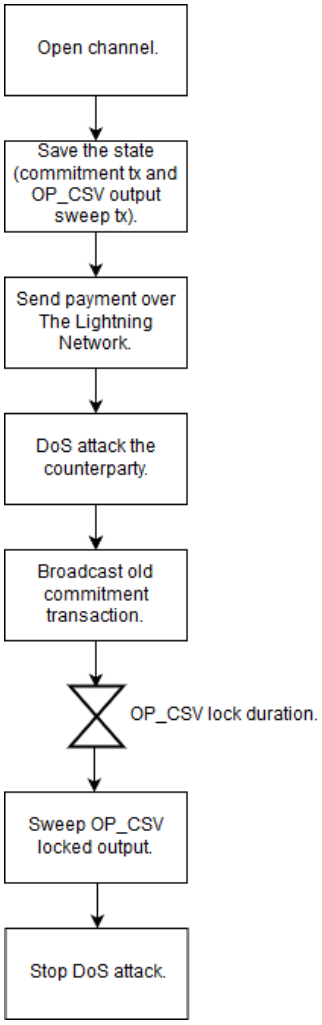

Figure 3: Generic version of the attack: the attacker manages to get an old Lightning Network channel state accepted into the blockchain by means of a DoS attack on the counterparty.

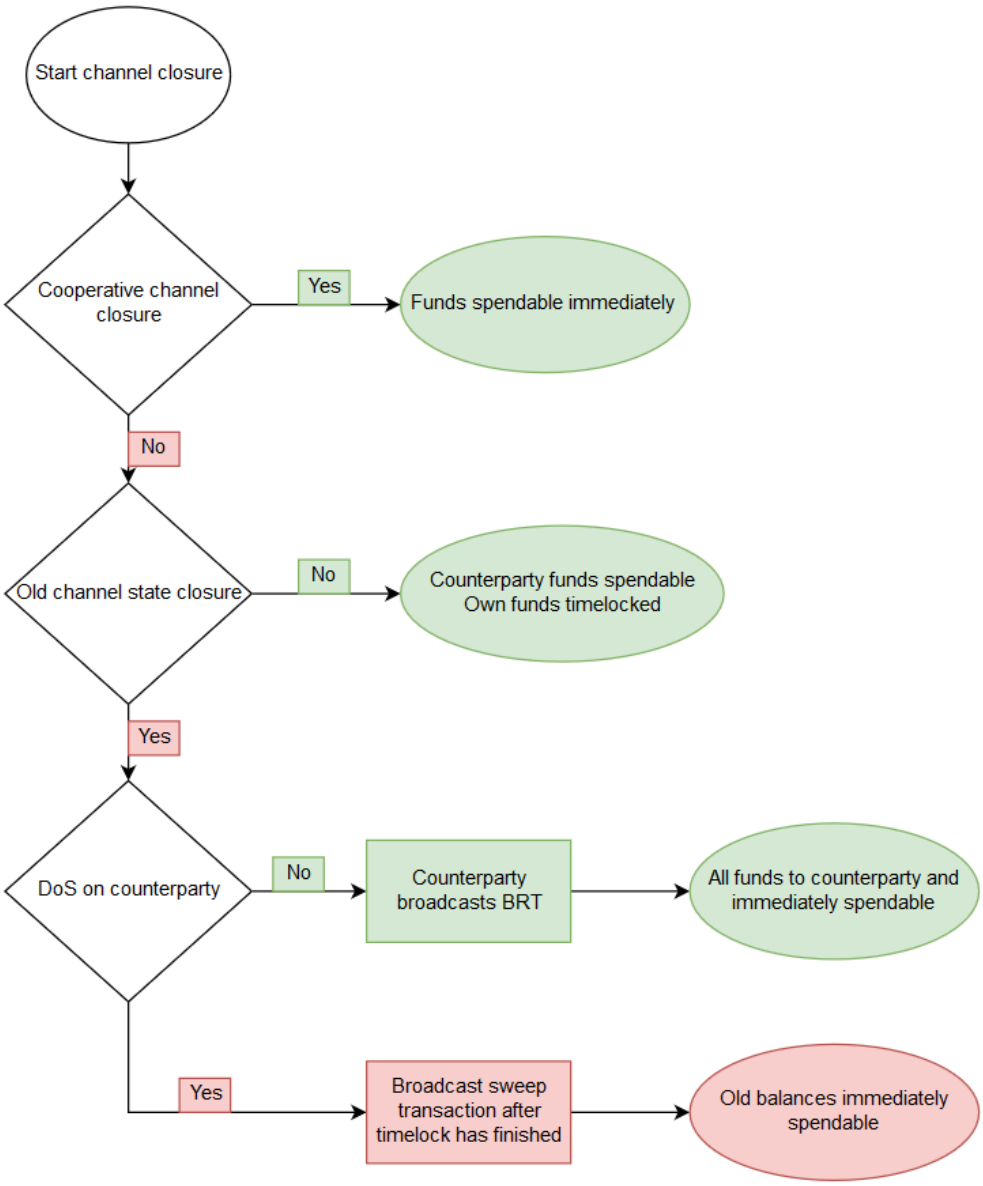

Figure 4: Flowchart of four different types of situations when closing a channel. Marked in red is the path an attacker would take.

6 Attack in practice

To execute the attack described in the previous section in practice using the LND, one has to differ from its intended protocol flow. The main challenge is getting access to an old channel's state Bitcoin transactions. Two ways to do so are provided in the next subsection. Furthermore, one needs to know the DoS attack duration and potential endpoints for this attack. Lastly, this subsection presents the results of a simulated attack.

6.1 Adjusting the LND

This subsection describes two methods of manipulating the attacker's LND, as such to make it possible to access an old channel state, and broadcast them to the blockchain. It is needed because by default its RPC layer or logging module does not provide a way to do so. The first method adjusts its source code to log the relevant transactions, after which the attacker is able to manually broadcast them at a later time. The second method initializes the LND with a manually outdated saved state.

6.1.1 Adjusting the source code

By adjusting the stateStep() method to not broadcast the transaction when doing an uncooperative channel closure, but merely log it in hexadecimal format, one is able to save an old commitment transaction and broadcast it in a later stage. This method can be found in contractcourt/channel_arbitrator.go[5] and this is accomplished by commenting out line 462 to 474 and adding the following lines at line 459:

```
buf := bytes.NewBuffer(make([]byte, 0, closeTx.SerializeSize()))
_ = closeTx.Serialize(buf)
log.Errorf("%x\n", buf.String())
```

Finally, one has to change line 478 from:

```
nextState = StateCommitmentBroadcasted
```

To:

```
nextState = StateDefault
```

One may now invoke the closechannel command, without in fact closing the channel. Note that one still needs the OP_CSV locked output sweep transaction to complete the attack. The necessary data to construct this transaction can be found in closeSummary.CommitResolution and may be logged by adding the following statement at line 450:

```
log.Errorf(spew.Sdump(closeSummary.CommitResolution))
```

The LND itself creates it in the method createSweepTx(), which can be found in utxonursery.go.

[5] https://github.com/lightningnetwork/lnd/blob/v0.4.2-beta/contractcourt/channel_arbitrator.go

6.1.2 Initializing the LND with an old state

An easier way is to backup a desired state by copying LND's complete data directory and initialize the LND using this backed-up data directory at a later point. The LND is then unaware it is in fact in an older state; so when invoking the closechannel command with the force parameter it will broadcast the old channel state and will sweep the OP_CSV locked output automatically when its timer has finished.

6.2 DoS duration

The needed DoS duration depends on the output of the commitment transaction that is time locked using OP_CSV. This is a relative time lock expressed in blocks, that must pass after the commitment transaction is confirmed on the blockchain, before the corresponding output can be swept. The LND scales this value linearly according to the channel value[11]. More specifically, it is calculated as a function of the min and max of OP_CSV lock time values and the max channel value accepted in satoshis. Those values are compiled by default to be respectively: 144 blocks, 2016 blocks and 16777215 satoshis (0.16777215 BTC). In the listing below, the function that defines the OP_CSV value in the LND for a given channel value can be found:

```
delay := maxOP_CSV * channelValue / maxValueAmount
if delay < minOP_CSV {
    delay = minOP_CSV
}
if delay > maxOP_CSV {
    delay = maxOP_CSV
}
```

Given the formula above, the following insights can be derived: the highest channel value to receive the minimum OP_CSV time of 144 blocks is 1206694 satoshis (0.01206694 BTC). Whereas, the max OP_CSV time of 2016 blocks is only reached at max channel value of 0.16777215 BTC.

Bitcoin aims to have a new block every ten minutes. By multiplying the OP_CSV value by ten minutes, and adding another 10 minutes for the output sweep transaction, one can derive the expected needed DoS duration. Consequently, the expected DoS duration for a channel value of 0.01206694 BTC is 145 blocks * 10 minutes, which equals to 24 hours and 10 minutes.

6.3 DoS targets

The DoS attack surface extends past the network layer and the LND, owing to the fact that the LND instructs a local Bitcoin node to broadcast the BRT in case of an attack. Thus crashing either the LND or the accompanying Bitcoin node is also sufficient for the attack to succeed.

6.4 Results

A lightning channel was funded by Mallory for 0.01206694 BTC and a payment of 0.01194182 BTC was sent to Alice by making use of this channel. An old channel state was broadcast by Mallory and it was included in block 1325791. The final sweep transaction to secure Mallory's funds was included in block 1325936. On average the attack would have taken 24 hours and 10 minutes, since this is the average time it takes to confirm 145 blocks. However, due to variety in block processing speed it took 43 hours, 3 minutes and 49 seconds. On the real Bitcoin network it is possible, but not likely that the expected and actual attack duration differs as much.

To summarise, Mallory successfully managed to get the initial state of 0.01206694 BTC (minus transaction fees) to Mallory accepted into Bitcoin's testnet3 blockchain, by making use of the technique proposed in section 6.1.2 together with a simulated successful DoS attack. Table 1 provides the relevant Bitcoin testnet3 transaction IDs.

Transaction type	**Transaction ID**	**In bitcoin testnet3 blockchain?**
Funding transaction	7d0d80c916fc956e555ae4d2bb516ac4 9dc8efbb990bb9427317d8e2e1bbba17	Yes
Old channel state commitment transaction	b06cc0d70d3a8faef975aab390c87027 4c9b963cde8d3754f1959f1874d620fc	Yes
OP_CSV output sweep transaction	099f1135d7ff29a318f31c45dff2b69e 7e3ea6971d2b68dd9a5974f8738a7e07	Yes
Latest channel state commitment transaction	ef5c8326cf522f0a4f30c818e8de8613 585b2768f22d5b98b6c29e7c3e99d726	No. Appendix section A.1 provides the signed transaction

Table 1: Relevant Bitcoin testnet3 transactions.

The latest channel state expressed as a commitment transaction signed by Alice and Mallory can be found in Appendix A.1. Note that there is no way to proof this was the latest channel state, however, it does proof that at *some* point the channel state was as such. Furthermore, it provides the interested reader with insight in the format of a commitment transaction. This transaction is now unusable, because it spends from the same inputs as the old channel state commitment transaction, which was broadcast by Mallory. Normally Alice would have been able to respond with the BRT, however, she was not aware and not able to broadcast it because she was under a DoS attack.

7 Discussion

The results show that in the Lightning Network it is possible to claim bitcoin that belongs to a counterparty by making use of a DoS attack. Whether an attack is worth doing so depends on many factors. Financially, it may depend on an attacker's cost to maintain a successful DoS attack for the needed duration.

This cost, expressed as an hourly upper limit for financial profitability can be estimated using the formula below. An attacker that is able to do a successful DoS attack on a lightning node, for an hourly price lower than the formula's outcome, is in theory able to do so profitably.

$$hourlyAttackerCostLimit = \frac{((bitcoinChannelValue * bitcoinPrice) - transactionFees)}{(144/(OP\_CSVlocktime + 1))/24}$$

Note that the LND has a linearly scaling OP_CSV value relating to the channel value. Therefore, the formula's outcome is the same for a channel value of e.g. 0.15 Bitcoin and 0.015 Bitcoin.

At the time of writing, at a price of $6700 per bitcoin the attack is profitable at an estimated attacking cost that is lower than $3.33 per hour. This value is an estimation due to the varying Bitcoin transaction fee costs. The calculation used an expected total fee cost of $1.50 for its needed three on chain transactions. However, even if an attacker's cost is higher than its direct return, attackers might still be motivated by indirect financial gain, e.g. by impacting the price of bitcoin. For instance, an attacker could short bitcoin, hoping a successful attack will result in a negative price reaction. Furthermore, a competing cryptocurrency that advertises a different scaling mechanism might comparatively increase in price.

A second factor the attacker has to take into account is the reliability of its DoS attack, since merely a short up-time is needed for the counterparty to detect that an old state has been broadcast and then reply with the BRT.

A solution to this attack might be what has been proposed as watchtowers, which are third parties that monitor the blockchain for old channel state transactions. It will need to receive all old channel states in order to function. To create a financial incentive an extra output can be added to the BRT that pays the watchtower. As a result the watchtower is eager to broadcast it, since it will be paid when doing so. It follows that for a successful attack, it is not enough anymore to merely DoS the channels counterparty, but one also has to attack its watchtower(s). Furthermore, one would need to figure out who the channel's counterparty watchtower is, which, if designed correctly, is non trivial. Admittedly, this is a solution that stops the attack demonstrated in this paper. However, Bitcoin's and the Lightning Network's promises are to provide a trustless system, to quote the Lightning Network paper authors: "An effectively **trustless** structure can be achieved by using time locks as a component to global consensus". So when one needs to rely on third-party watchtowers to use the system safely, it can be argued that the system is not trustless anymore, which opposes the Lightning Networks own design goals. While it is possible to host ones own watchtower, this will impact the property of decentralization, since the cost of running a lightning node increases.

A new mechanism called Eltoo for updating channel states has been proposed by Blockstream, which in the future might be merged into the Lightning Network. It makes use of floating transactions, which allow for spending a predecessor's transaction output. This means that the latest agreed upon state and

thus Bitcoin transaction will be able to spend the outputs from an earlier state. A benefit it provides over its current design is that intermediary states do not have to be saved. Likewise, watchtowers benefit from the same property, since now they only need to store the latest channel state to remedy an incorrect broadcast. However, in light of the attack proposed in this research, it actually decreases an attacker's risk. Unlike in a lightnings channel current design, it does not provides a BRT like punishment transaction. Therefore, the attacker's risk becomes limited to losing transaction fees when broadcasting an old state.

8 Conclusion

The aim of this research was to investigate whether it is possible to steal bitcoin by the use of a DoS attack on a lightning channel. After investigating the mechanism used to invalidate old channel states, an attack was presented to do so. This work then successfully executed this attack in practice using the LND. Even though during the attack the DoS attack was merely simulated by disabling Alice's networking interface, a real uninterrupted DoS attack on Alice would of achieved the same results.

In addition, we provided a formula that estimates the maximum hourly cost to make this attack profitable. At a bitcoin price of $6700, it estimates that a DoS cost lower than $3.33 per hour will net the attacker a profit. However, for profit attacks as presented in this research have not yet been seen in the wild. A reason may be that the hard coded channel value limits are sufficient to keep financially motivated attackers away. Or, because the services and goods one is able to buy over the Lightning Network is limited to a few early adopters. When constraints on channel values are lifted, and adoption increases, the attack presented in this paper may become a serious risk. Despite of the findings in this work, it is a fact that Lightning Network's software is still in beta. Whether the Lightning Network will grow into a safe and widely used payment system remains to be seen.

9 Future Work

Instead of simulating a DoS attack by disabling its victim's networking interface, it would be interesting to explore DoS attack vectors on both the application and network layer. If it is able to provide a cost structure for an attack, taking into account the target's machine and network capacity, it will help set adequate OP_CSV values. Additionally, to investigate whether it is possible to steal bitcoin from a lightning node to which one is not directly connected, one should investigate HTLCs in a multi-hop channel context on the Lightning Network.

References

[1] Poon, Joseph, and Thaddeus Dryja. "The Bitcoin Lightning Network: Scalable off-chain instant payments." draft version 0.5.9.2 (2016), https://lightning.network/lightning-network-paper.pdf

[2] The Bitcoin Lightning Network summary, https://lightning.network/lightning-network-summary.pdf

[3] The Bitcoin Lightning Network technical summary, https://lightning.network/lightning-network-technical-summary.pdf

[4] Peter Todd's criticism on the Lightning Network. Retrieved May 13, 2018 https://twitter.com/peterktodd/status/968190536812236802, 2018.

[5] bitPico DDoS against mainnet Lightning Network. Retrieved May 16, 2018 https://twitter.com/bitPico/status/980978688740163584

[6] Lightning Network DDoS Sends 20% of Nodes Down. Retrieved May 16, 2018 https://www.trustnodes.com/2018/03/21/lightning-network-ddos-sends-20-nodes

[7] McCorry, P., Mser, M., Shahandasti, S. F., Hao, F. "Towards bitcoin payment networks." Australasian Conference on Information Security and Privacy. Springer, Cham, 2016.

[8] Hashed Timelock Contracts, https://en.bitcoin.it/wiki/Hashed_Timelock_Contracts

[9] Decker, Christian, Rusty Russell, and Olaoluwa Osuntokun. "eltoo: A Simple Layer2 Protocol for Bitcoin, 2018, https://blockstream.com/eltoo.pdf

[10] Commitment Transaction Outputs, Retrieved june 2018, https://github.com/lightningnetwork/lightning-rfc/blob/master/03-transactions.mdto_local-output

[11] Relative time lock scaling, https://github.com/lightningnetwork/lnd/blob/v0.4.2-beta/lnd.go

A Appendix

A.1 Latest channel state during attack simulation


```
{
  "txid": "ef5c8326cf522f0a4f30c818e8de8613585b2768f22d5b98b6c29e7c3e99d726",
  "hash": "1f209b77e7a9261022e54108c1c126abd1d08e48e0ed4da77965f2affd2dac93",
  "version": 2,
  "size": 345,
  "vsize": 180,
  "locktime": 542332097,
  "vin": [
    {
      "txid": "7d0d80c916fc956e555ae4d2bb516ac49dc8efbb990bb9427317d8e2e1bbba17"
      "vout": 0,
      "scriptSig": {
        "asm": "",
        "hex": ""
      },
      "txinwitness": [
        "",
        "304402207d9cfb87666014b2f5e481a4521b51544c8220378051f9f68b8335f9874bb6
        5202202572e996c4c3d6d90053583bb73062db8ac5eb4aa41fc0251c0478631f8e6c930
        1",
        "304402206ec9f1df33fa7d5fdab3d7a9a8ca2cdf082eea03062c12d4cf043932d10a0a
        28022033789f5e5d3b9cf90134c49388077c0c74ea2f3c5e75cbeae679aaa75eb6be700
        1",
        "522102517ee377a51bfb14d74d0bf33e6ce779974612ece214e96b8bea6697b3c78a65
        21026aff4c05426e256485a90c027ce8fd0f066cee8ffe4dd93d36abb8728121c7f752a
        e"
      ],
      "sequence": 2148172452
    }
  ],
  "vout": [
    {
      "value": 0.00012512,
      "n": 0,
      "scriptPubKey": {
        "asm": "0 24e5510b66924ce4dfd66d4b5996eea1a37904aac387ba9db657bd06d452d
        12c",
        "hex": "002024e5510b66924ce4dfd66d4b5996eea1a37904aac387ba9db657bd06d45
        2d12c",
        "reqSigs": 1,
        "type": "witness_v0_scripthash",
        "addresses": [
```

```
            "tb1qynj4zzmxjfxwfh7kd494n9hw5x3hjp92cwrm48dk277sd4zj6ykqwcaesc"
          ]
        }
      },
      {
        "value": 0.01194001,
        "n": 1,
        "scriptPubKey": {
          "asm": "0 72606f0ba607eaeed3c37f8822628a680bed384f",
          "hex": "001472606f0ba607eaeed3c37f8822628a680bed384f",
          "reqSigs": 1,
          "type": "witness_v0_keyhash",
          "addresses": [
            "tb1qwfsx7zaxql4wa57r07yzyc52dq976wz0zhr6z0"
          ]
        }
      }
    ]
}
```