Stressing Out: Bitcoin "Stress Testing"

Khaled Baqer[1], Danny Yuxing Huang[2], Damon McCoy[3], and Nicholas Weaver[4]

[1] Computer Laboratory, University of Cambridge
[2] University of California, San Diego
[3] New York University
[4] International Computer Science Institute

Abstract. In this paper, we present an empirical study of a recent spam campaign (a "stress test") that resulted in a DoS attack on Bitcoin. The goal of our investigation being to understand the methods spammers used and impact on Bitcoin users. To this end, we used a clustering based method to detect spam transactions. We then validate the clustering results and generate a conservative estimate that 385,256 (23.41%) out of 1,645,667 total transactions were spam during the 10 day period at the peak of the campaign. We show the impact of increasing non-spam transaction fees from 45 to 68 Satoshis/byte (from $0.11 to $0.17 USD per kilobyte of transaction) on average, and increasing delays in processing non-spam transactions from 0.33 to 2.67 hours on average, as well as estimate the cost of this spam attack at 201 BTC (or $49,000 USD). We conclude by pointing out changes that could be made to Bitcoin transaction fees that would mitigate some of the spam techniques used to effectively DoS Bitcoin.

1 Introduction

The Bitcoin network [9] was subjected to a major spam campaign during the summer of 2015 that caused degraded performance of Bitcoin. The likely intent of the incident (advertised as a "stress test") was to Denial of Service (DoS) Bitcoin with spam transactions, in order to expose the vulnerability of Bitcoin to spam attacks and to garner support for a proposed change to increase the number of transactions that the Bitcoin network can verify, which is currently approximately 3 transactions per second. DoS attacks against Bitcoin have been theorized. However, to date there has been little empirical analysis of DoS attacks launched directly against Bitcoin.

In this paper, we conduct an empirical analysis of this spam based DoS attack launched against Bitcoin. To enable our analysis, we use k-means clustering and a set of features we identified to differentiate spam from non-spam transactions. We validate the results of our clustering technique and are able to identify that 385,256 (23.41%) out of 1,645,667 total transactions were spam between July 7^{th} and July 17^{th}, which corresponds to the peak of the spam based DoS attack. Further analysis of transactions in these clusters allowed us to identify four distinct motifs of spam transactions. Based on our identification of spam and

non-spam transactions we are able to measure the cost of this spam campaign and impact on non-spam transactions in terms of delay and increased fees.

Our study makes several contributions, including proposing and empirically validating a method to identify spam transactions, characterizing the spam transactions, and measuring the impact of this spam campaign on Bitcoin. Finally, in our discussion section we propose changes to transaction fees that would mitigate the effectiveness of DoS attacks that use spam motifs similar to those used in this attack.

2 Background

Bitcoin transactions are chained signed receipts, consisting of one or more signed inputs to spend, and one or more outputs. The outputs of the transaction are normally assigned to Bitcoin addresses; the hash of a public key that has the authority to use the particular output as an input to another transaction. Transactions are included in *blocks*, with each block also including the hash of the previous block to create a *blockchain*. A block results from verifying all included transactions, with a hash of the data creating a digest with a network-determined prefix of zeros. The latter constitutes the difficulty of the network which is automatically tuned to ensure that the network expects that each block takes 10 minutes to create, and the effort exerted to create the correct digests is Bitcoin's Proof-of-Work (PoW). The blockchain represents Bitcoin's global ledger, and miners compete to create blocks and broadcast them to the network to claim their rewards. Currently the network only creates and accepts blocks of 1 MB or less, limiting global transaction rate to less than 3 transactions per second.

The main components of a transaction, relevant to our analysis, are the transaction ID (*txid*), the inputs to the transaction (*vin*), and the outputs (*vout*). A transaction includes inputs that reference outputs of one or more older transactions. That is, each input includes, *inter alia*, a reference to an older transaction and the index in the list of outputs (of the referenced transactions) to be used. Bitcoin transactions vary in their inputs and outputs, which determine the size of a transaction.

Transactions are broadcasted to other peers in the Bitcoin P2P network, who perform local verifications to prevent DoS attacks, and the transaction propagates the entire network within a few seconds [3]. Received transactions are maintained in a node's own local memory pool (*Mempool*). Here, transactions remain in limbo until confirmed and included in a block; once a transaction is included in a block, a node removes the transaction from its Mempool. Although a node tends to maintain unconfirmed transactions for a very long period of time, memory pressure may cause a node to evict old entries from the Mempool if it grows sufficiently large.

Nodes also maintain an unspent transaction output set (UTXO) to easily verify inputs to newly received transactions. Therefore, an increase in the UTXO adds memory pressure on nodes which currently hold the UTXO set in RAM.

Unlike the Mempool, memory pressure on the UTXO set cannot be relieved by eviction, but requires changing the node's implementation.

In the reference implementation, a Bitcoin miner calculates a priority and uses this to determine which transactions to include in the block. To calculate transaction priority (P), the node considers all inputs to the transaction as well as its size. P is defined in Bitcoin as $\sum_{i=0}^{n}(value_i \times age_i) \div S$, where n is the number of inputs to the transaction, $value$ is the value of input i (in Satoshis[1]), age is defined as the difference between the current block's height and the input's block height, and S is the transaction's size. The value of P determines a transaction's fate; there are three possibilities:

1. Include transactions in the high-priority section of a block (50 KB); no transaction fee is necessary. The following conditions must be satisfied, the transaction must be:
   - smaller than 1 KB
   - all output values are at least 0.01 BTC
   - P is high as determined by $value_i$ and age_i
2. Transactions that pay fees are prioritized by highest mBTC per KB.
3. The remaining transactions are maintained in the Mempool until one of the two conditions above is satisfied.

In the latter case, age is the determining factor for P since everything else is constant. It's of particular note that miners prioritize for higher fees.

2.1 DoS Targets inherent in Bitcoin

Spam can be detrimental to the Bitcoin network by outcompeting legitimate transactions for inclusion in a block, delaying other transactions. We define the following types of spam:

1. **Fan-out**: Transactions that split a few inputs into many outputs occupy space in the blocks and also increase the UTXO set.
2. **Fan-in**: Transactions which absorb a large number of inputs reduce the UTXO set but still occupy substantial space in the blocks.
3. **Dust output**: Transactions that create very small "dust" outputs convey a trivially small amount of value but occupy the same amount of resources in the Bitcoin network.

The spam campaigns in the "stress test" target one or more aspects of the Bitcoin environment, including the block size limit, the UTXO set, and the computational cost for verification. All these limited resources represent potential targets.

The primary publicly stated motivation behind the stress test campaign was to provide a justification for raising the Bitcoin block size limit before organic

[1] 1 Satoshi = 10^{-8} bitcoins. We follow the convention of referring to the protocol as *Bitcoin*, the currency and its units as *bitcoin* or BTC.

demand limits the ability of Bitcoin to process payments. The current Bitcoin block size of 1 MB globally supports less than 3 Bitcoin transactions per second. Since this is three orders of magnitude lower than Visa's sustained rate of 150M transactions per day (and peak processing ability of 24,000 transactions per second) [10], it's clear that the current Bitcoin payment processing is insufficient to meet the ambitions of the Bitcoin community. The public intent was to demonstrate the impact of this limit by squeezing out normal transactions.

Raising the block size, however, opens up a different DoS vulnerability: a long term growth DoS on the Blockchain itself. Since the Blockchain records all previous transactions, an attacker could perform low fee transactions simply to consume space. Thus if Bitcoin raised the block limit to 20 MB, and an attacker can cheaply consume 10 MB of data per block, this causes the Blockchain to increase in size by half a terabyte a year.

Since valid transactions can only spend unspent outputs, most full Bitcoin nodes keep the UTXO set in memory to speed transaction validation. The memory requirements for the UTXO set are solely based on the number of unspent outputs, so the inclusion of dust outputs in the stress test adds memory pressure to the UTXO set. A better designed Bitcoin node should not have this vulnerability.

Another DoS attack occurred on October 7^{th} and 8^{th}, which also put a significant amount of pressure on the Mempool memory, raising the Mempool to nearly a GB, with a transaction backlog of nearly a week. Since there are a large number of nodes running on Raspberry Pi and other constrained systems, this large Mempool managed to crash over 10% of all Bitcoin nodes [2]. Most of the spam itself, however, was of low priority. Such spam does not put pressure on block inclusion, but neither does it cost the spammer any bitcoins; transactions that are never confirmed do not incur a cost for the sender.

An inadvertent CPU DoS occurred due to a mining-pool's "cleanup" block, a single 1 MB transaction that served to remove a massive number of unspent transactions sent to crackable "Brain wallet" addresses (which use a passphrase, instead of private keys, to create Bitcoin addresses and spend bitcoins). Other nodes required substantial CPU time to validate this block, as the current implementation required $O(n^2)$ time to validate a transaction. There may be other CPU DoS possibilities inherent in the Bitcoin protocol that attackers can exploit.

Another DoS is inherent in "transaction malleability". Someone can take a valid transaction, permute it so it has a different *txid*, and broadcast that modified transaction to the network. If the attacker's transaction is accepted into the blockchain, this can disrupt wallet services, hardware wallets, and other systems tracking *txid*s to determine when a transaction commits to the blockchain. Recently, an attacker performed this DoS "because I am able to do it." [3]

Finally, a later (failed) spam campaign attempted to flood the network with invalid transactions, perhaps intending either a traffic DoS or a CPU DoS. The

[2] `https://www.reddit.com/r/Bitcoin/comments/3ny3tw/with_a_1gb_mempool_1000_nodes_are_now_down`

[3] `https://bitcointalk.org/index.php?topic=1198032.msg12579271`

"money drop", a public release of private keys by one of the purported instigators of the stress test, seems intended to cause a big race which would cause a large number of "double-spend" transactions. This did not produce a meaningful disruption of the network, although it was probably intended to introduce computational load.

One aspect not encountered during the stress test was the effect of filtering valid but spammy transactions. The introduction of spam filters, if an unknown attacker continued a longer term DoS attempt, could in itself be a DoS. If the attacker adapts to the filters, eventually the filters will either fail to stop the spam or incur false positives. Even a small false positive rate might be disruptive: could a payment network tolerate a 1-2% transaction failure rate due to spam filters?

3 Data collection

In our study, we set up a server connected to a public-facing network. We installed Bitcoin Core 0.11 and kept it running between June 19 and September 23, 2015. We collected three main data sets using Bitcoin daemon's JSON-RPC interface.

1. Bitcoin Blockchain: On September 23, we downloaded the entire blockchain using the `getblock` and `getrawtransaction` methods. This returned details for all blocks and transactions, such as the timestamps of blocks, the timestamps at which we received the transactions, the number of transaction inputs and outputs, as well as the input and output amount. We stored the data as plain-text JSON strings. As a result, the total data size is 350 GB.
2. Mempool: Between June 19 and September 23, the `getrawMempool` method was invoked every minute. This returned a list of unconfirmed *txid*s currently in the Mempool. These would be either committed to the blockchain or later discarded by the P2P network. We saved this list of *txid*s, along with the timestamp of the RPC call, on the Hadoop file system. During this period, we captured 12 million distinct *txid*s in the Mempool, which amounts to 250 GB of plain-text data.
3. Unconfirmed transactions: For every unconfirmed transaction that we had obtained above, we *immediately* looked up the transaction details using the `getrawtransaction` method, since the Mempool could discard the transaction any moment. To optimize for speed and storage, we ignored transactions that we had previously seen. Finally, we saved all the transaction details, along with the data collection timestamp, on Hadoop. Between June 19 and September 23, we captured 1.3 TB of unconfirmed transactions in plain text.

The total size of the data collected is 2 TB, which we saved as plain-text JSON strings on the Hadoop file system and analyzed with Spark. We summarize our data sets in Table 1.

As we collected data using only a single node, our perspective of the P2P network—and thus the transactions in the Mempool—is potentially biased. In particular, network propagation takes time. For transactions in the Mempool,

Table 1. Data sets. All data sets cover a period between June 19 and September 23.

Data	Period	Size
Blockchain	Between Jan 9, 2009 and Sept 23, 2015	350 GB
Memory pool	Between June 19 and Sept 23, 2015	250 GB
Unconfirmed transactions	Between June 19 and Sept 23, 2015	1.3 TB

the timestamps that we observed may be later than the originating timestamps. Furthermore, whether a transaction is relayed is up to individual nodes. A transaction created a few hops away is not guaranteed to reach our node. It is, however, beyond the scope of this paper to adjust for such biases. We assume that our observation of the network is largely consistent with the rest of the network.

4 Spam clustering

We use an unsupervised machine learning method, k-means clustering, to find similarities and evaluate our findings. This is not necessarily a perfect filter, but as we manually verify, this does efficiently detect the spam transactions in the "stress test".

To use k-means clustering, we create a multi-dimensional vector representing features of a Bitcoin transaction. We include in Table 2 the list of features and follow up with defining features that were not previously discussed.

Table 2. Transaction features

Feature	Notation	Description
Inputs	I	Number of inputs
Outputs	O	Number of outputs
Ratio	R	$I \div O$
Priority	P	Value-weighted measurement
Size	S	Size (bytes)
Size and Ratio	$S \times R$	Emphasize fan-in and fan-out
Fees	F	Value of unclaimed outputs
Coin days destroyed	CDD	Coin age and spending velocity
Value	V	Total output value
Fees to values ratio	$F \div V$	Emphasize fee differences

R is necessary to highlight the difference between fan-in and fan-out transactions. We further highlight this difference by multiplying the size of the transaction by its ratio (otherwise, transactions with clear differences in R are clustered together based on similarities in S). We include another property to highlight the velocity of spending bitcoins represented as CDD[4]. This feature gives more

[4] This feature is used by Bitcoin block explorers, see for example: `https://blockr.io`

weight to older coins, and can be calculated as $\sum_{i=0}^{n}(value_i \times age_i)$. Unlike P, *CDD* does not consider S, *age* is measured in number of days rather than blocks (an estimate of 144 blocks are produced each day), and *value* is in bitcoins.

4.1 Methodology

Since spam campaigns may not link transactions and addresses together, parsing the blockchain to look for linked transactions might be a futile process. Our approach is different: we cluster transactions based on their motifs (trends in the Bitcoin network), and disregard transactions' identifying information (output addresses, *txid*, etc.). Our main assumptions at this stage echo those required for machine learning algorithms: a pattern exists, we cannot mathematically point out differences in patterns (without data visibility), and we have a large trove of data to show the patterns exist. We assume motifs do exist because spam requires construction in-bulk to have a measurable effect on the network. Thus spammers naturally create large numbers of transactions that "look similar". We also expect that such groups of transactions may have different motifs compared with normal Bitcoin behavior, since spammers want to minimize the cost and maximize the impact, producing different types of transactions (e.g. very high fan-out or dust output) that particularly stress the network.

What we seek is a high-level interpretation of the data into distinct clusters that we can then use to label transactions as spam and validate our results. Thus, to investigate our main goal of identifying spam motifs, we consider the entire Bitcoin network as an entity, rather than analyzing features of a transaction independently from network norms. The latter process relies heavily on what features should be considered to identify spam, which might assign more weight to some features while disregarding others that are more influential.

We use k-means clustering, as provided in Spark's machine learning library (`MLlib`). k-means clustering is a type of machine learning algorithm for unsupervised learning. This algorithm is particularly useful to cluster similar data together when it is non-trivial to define *similarity* using the unlabeled data. Similarity of vectorized data is determined using k-means by minimizing the Within-Cluster Sum of Squares (WCSS); the data is matched to the cluster centroid with the closest mean. The following equation is used to iterate over the data to get optimal cluster centroids in order to minimize WCSS: $min \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2$, where k is the number of clusters, x is the data element (in vector form), S_i is the set containing n elements, and μ_i is the mean of S_i (i.e. the mean of all the elements in vector form that are contained in S_i).

To reproduce the results discussed in this paper, the following properties of k-means must be considered: the number of clusters k was set to 10, the number of `maxIterations` was set to 100, and `initializationMode` was set to `random`. The silhouette coefficient measures the homogeneity of the data in a cluster. This is performed by measuring the average dissimilarity (defined in terms of distance between data elements) between a given element within its cluster, and comparing the result with the average dissimilarity between that

same element and elements of another cluster considered to be the next best-fit. However, in our case, our aim is to show general transaction motifs, rather than to show detailed transaction differences or find anomalies. We arrive at $k = 10$ after testing multiple values for k to show enough visibility of transaction patterns. If we choose $k = 11$ for example, we obtain a new cluster where the average of transaction outputs is 8 rather than 11 (as shown in cluster 9 in Table 3). Instead, we accept that the clustering algorithm groups these transactions together in cluster 9, given that they are similar in other features. Conversely, with $k < 10$, clusters contain transactions that differ in most of their features; this does not enable us to inspect the clusters to easily determine which of them fit our definitions of spam. With $k = 10$, we see the "outliers" visible in a dedicated cluster (cluster 8 in Table 3), whereas with $k < 10$ these outliers are included in other clusters that do not match well.

The initial step for processing data was weeding out some transactions that alter the clustering results. To set a starting point, we create two checks to filter transactions. First, we check if the transaction creates dust output (we explain this check in details later). The second check determines if the transaction's fan-out ratio is unusual (a threshold is set at 0.3). The rationale for these two checks is as follows: If a fan-in transaction creates dust output, then it qualifies as spam, otherwise it is minimizing the set of UTXOs that must be maintained to verify transactions. Moreover, if a fan-out is unusual, this is enough to qualify a transaction for clustering, and we later determine if the transaction is spam by inspecting clustering results, and checking for dust outputs in clusters that seem to contain normal transactions.

We analyze confirmed transactions that occurred between June 24^{th} and July 17^{th}, 2015. The total number of transactions in this epoch is 3,321,429. To obtain k-means clusters, we perform k-means training on all transactions that were confirmed during the July spam campaign epoch, that occurred between July 7^{th} and 17^{th}, the total number of transactions in this training epoch is 1,645,667. Using the cluster centroids from the spam epoch, we analyze the pre-spam epoch to validate our results.

4.2 Results and motifs

We now discuss motifs found in more than 1.6M transactions that occurred during the spam epoch. Table 3 shows each cluster centroid's features. As discussed earlier, these centroids are the result of optimizing WCSS, and are represented as the means of the values of all transactions in the corresponding cluster. Table 4 shows the standard deviation of the cluster centroids[5].

1. **Fan-in. Clusters 2 and 4** include about 14K fan-in transactions. The pattern is distinct: large I and one O (in rare cases O is for two addresses).

[5] The notation used in the tables corresponds to the notation used for the transaction features defined earlier. Note that both tables include rounded values, while attempting to maintain distinctions for small values with the minimum amount of rounding necessary. For better presentation, we omit some features.

Table 3. Cluster centroids (*confirmed transactions*)

C	*TXs*	*I*	*O*	*R*	*P*	*S*	*F*	*CDD*	*V*
0	48K	1.35	46	0.06	0.74	1.8K	0.0004	0.195	4.06
1	28	4.4K	1	4.4K	0.001	645K	0.04	0.06	0.0
2	896	106	1	103	0.17	16K	0.001	0.34	0.13
3	20	1.1K	1	1.1K	0.0008	162K	0.01	0.012	0.0
4	13.5K	31	1	31	0.04	4.7K	0.0002	0.02	0.006
5	16	1.4	13	0.15	535K	668	0.0004	25K	1K
6	9.5K	20	17	19	0.4	3.5K	0.0004	0.14	1.4
7	425K	1.1	2	0.8	1	224	0.0001	0.022	1.43
8	2	1	19	0.05	136M	787	0.0002	740K	3K
9	117K	1.2	11	0.14	72.43	561	0.0002	2.7	6.5

Table 4. Standard deviation of selected features (*confirmed transactions*)

C	*I*	*O*	*R*	*P*	*S*	*F*	*CDD*	*V*
0	4	104	0.77	27	3.6	0.002	17	40
1	1.2K	0	1.2K	0	176	0.012	0.05	0
2	43	0.2	35	2	6	0.0005	4	1.8
3	403	0	403	0	60	0.004	0.02	0
4	8	0.1	8	0.8	1.2	0.0002	0.5	0.24
5	1	7	0.1	0.35M	0.38	0.0001	26K	1.2K
6	2	0.4	2	1.65	0.35	0.0002	0.5	4
7	0.4	0.9	0.4	9	0.1	0.0002	0.2	15
8	0.0	0.5	0	3M	0.02	0	0.2M	748
9	0.5	6	0.2	2K	0.2	0.9μ	177	70

The transactions vary in S due to variations in I, and a notable distinction is in *CDD*. Cluster 2 includes larger values for *CDD*, which indicates that the inputs are not used for rapid transfer of value. Moreover, these transactions may not have been used as spam *per se*, but are rather part of tumblers or mixers where a large number of inputs are collated into single outputs and the chain continues, in order to mix coins together and obtain relatively better privacy. These transactions involve long chains of many inputs to a single address, the last address then transfers funds to multiple outputs in fan-out transactions, and so on. A large number of fan-in transactions impact the Mempool, but minimize the UTXO set.

2. **Fan-out.** The fan-out pattern involves one or two addresses sending funds to many addresses, as shown in **Clusters 0, 5, 8 and 9**; the total number of transactions in these clusters is about 165K. These transactions increase the UTXO set. This pattern was dominant in the clustering results; it resulted in multiple clusters for fan-out transactions that differ in features other than R. A low value for *CDD* indicates a fast movement of coins. Note that Cluster 0 includes transactions that have a single address sending small amounts to more than 3K addresses.
3. **Unable-to-decode.** With 425K transactions, **Cluster 7** includes the largest number of transactions. The distinct feature of most of these transactions is a one-to-one mapping: one address sending to a single output that cannot be decoded. Moreover, the fees paid for these transactions (which are collected by miners since the output cannot be decoded) equal the default fee value

of 0.1 mBTC per KB. Another feature of this cluster is the zero value for *CDD* (and low P), which indicates rapid movement of bitcoins.

4. **Dust.** The final motif of the analyzed spam campaign is the dust transactions we had previously discussed. **Cluster 7** contains non-spam transactions; normal transactions are matched to this cluster since they look similar to unable-to-decode transactions (low values for most features). It is not straightforward to visually inspect the cluster samples and determine if they are indeed spam. Therefore, we parse the transactions in this cluster to determine which of them fit our definition of dust spam. We explain in a later section how we parse the results to find dust spam transactions.
5. **UTXO cleanup. Clusters 1 and 3** include 'clean-up' transactions, created by miners to collate spam transactions to minimize the UTXO, thereby decreasing the spam impact on the network. The output addresses value of these transactions may be zero, meaning that all the inputs are collected as fees by the miner who includes the transaction in a block. Clean-up transactions include 'Brain wallet' addresses (discussed earlier). These two clusters are not categorized as spam, and the transactions are a consequence of the spam campaign. The number of inputs to these transactions range between 1K and 5K (resulting in a large standard deviation).

Note that clusters 5 and 8 contain few transactions due to their unusually high P. Cluster 8, which contains only two transactions, is indeed interesting and earns its unique cluster: along with high P, the values of these transactions are around 2,500 and 3,995 bitcoins (that is almost $0.6M and $0.96M in USD respectively). Both transactions include a generous fee of 0.002 BTC.

In summary **Clusters 0, 2, 4, 6, 7, and 9** correspond to our definition of Bitcoin spam, including dust transactions and unusual ratios, while clusters 1 and 3 are a consequence of spam and not spam motifs.

4.3 Validation

It is important to note that we lack an external source to create ground truth for our results. Without a labeled data set, or a third-party spam list, we cannot measure the clustering results to be spam more accurately than matching the results to our definitions of spam.

In order to find dust transactions, we check if P is low (less than 57M) and whether the transaction creates any outputs of 0.1 mBTC (about $0.02), which is the default fee value. We consider this a conservative estimate of the dust transactions involved in the spam campaign, and at the same time we consider the 0.01 BTC normally involved in dust checks to be too large.

We also applied clustering to transactions that occurred in the pre-spam epoch, between June 24^{th} and July 7^{th} (after filtering for dust and unusual ratios). The results are discussed in the next section, where we see a difference in the intensity of motifs before and during the spam epoch. This validates our clustering results: we find that the centroids obtained from training k-means, using the spam epoch data, can also detect spam patterns in non-spam epochs.

5 Impact on Bitcoin

We now describe the effects of spam campaigns on the Bitcoin network—especially on users who send non-spam transactions, as well as the miners. For the users, we measure the change in transaction fees and transaction delays (i.e. the time between when we first observe a transaction in the Mempool and when the transaction is committed to the blockchain). A large amount of spam is likely to increase the backlog of unconfirmed transactions. As a result, transactions are delayed for longer time periods. With more intense competition, senders pay higher fees, in the hope that their transactions will be included in blocks sooner. For the miners, we measure the corresponding increase in the block reward.

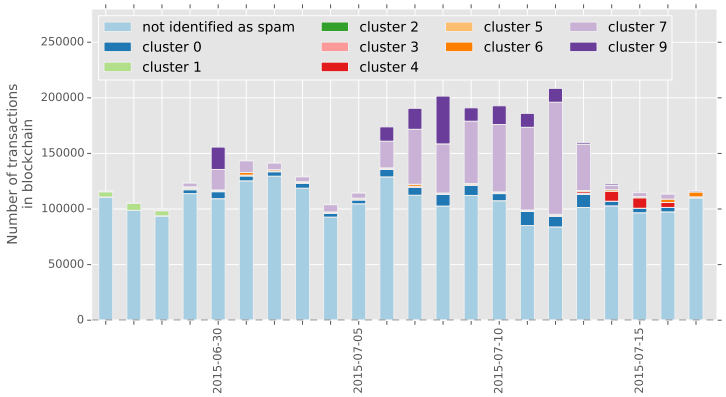

Fig. 1. A stacked bar chart that shows the number of transactions per day in the blockchain. Note that the spam period is from July 7^{th} to 17^{th}.

Figure 1 shows the clustering results in the non-spam and spam epochs. Note that in the pre-spam epoch (before July 7^{th}), clustering results show Cluster 1 transactions (UTXO-cleanup motif). This does not mean that miners were cleaning up spam; these transactions are similar to UTXO-cleanup transactions in terms of high I and low O, and similar P values.

To highlight periods of the spam campaign, we measure the number of unconfirmed transactions in the Mempool, which indicates the amount of backlog in the network. Every minute, we take a snapshot of the Mempool and count the number of unconfirmed transactions. We take the average on a daily basis and plot the result in Figure 2.

Each major spike in the graph refers to a period of significant backlog. The first spike, which happened between July 7^{th} and 17^{th}, corresponds to the spam campaign in our study. There are sporadic spikes between July and August, but we do not have sufficient insight on the cause. Finally, a spike appeared

around September 13, when an anonymous group conducted another stress-test on the network with their "money drop" (as discussed earlier). As a result, a large number of transactions were created to compete for the free bitcoins, although only a few of them would be included in the blockchain eventually. Such a deluge of transactions caused the second backlog in Figure 2. We do not, however, consider these transactions as spam.

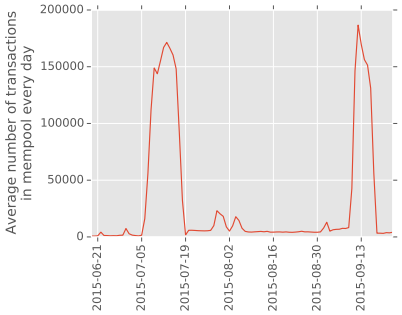

Fig. 2. The average number of unconfirmed transactions in the memory pool every day.

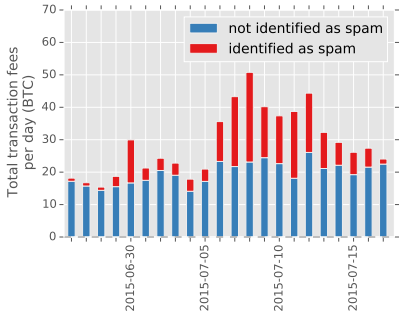

Fig. 3. A stacked bar chart showing the total amount of transaction fees every day.

Focusing on the mid-July spam period, we next examine the number of transactions that were committed to the blockchain. We are interested in how each block allocates its scarce 1 MB of real-estate space to spammers and non-spammers. As shown in Figure 3, the number of transactions surged during the spam epoch. Between a quarter to half of the daily transactions have been identified as spam. As a baseline comparison, we also show that the number of spam transactions before the spam period is significantly lower, with the exception of June 30. Based on anecdotal evidence, some users were attempting to stress-test the Bitcoin network on a small scale, which resulted in a brief rise in spam.[6]

For the non-spammers, the spam period was a time when both transaction fees and delays were higher than normal. We show the comparison in Figures 4 and 5. On average, the delays in processing non-spam transactions increases by 7 times, from 0.33 to 2.67 hours. Likewise, the average non-spam transaction fees also surged, increasing from 45 to 68 Satoshis for every byte of transactions (or from $0.11 to $0.17 USD per kilobyte of transactions)—an uptick of 51%.

While non-spammers suffered, miners slightly benefit from the fee hike. As shown in Figure 3, miners were earning twice their normal fee-based revenue during the mid-July spam period, as compared with the non-spam period. However,

[6] `http://motherboard.vice.com/read/wikileaks-is-now-a-target-in-the-massive-spam-attack-on-bitcoin`

even on days with maximum fees, the amount of extra mining income from fees was less than 1% of the block reward (which is 25 BTC per block, and about 3,600 BTC per day). The total transaction fees that spam transactions paid amounted to 201 BTC (or about $49,000 USD) over a 10 day time period—a modest sum that caused a rather noticeable disruption to the network.

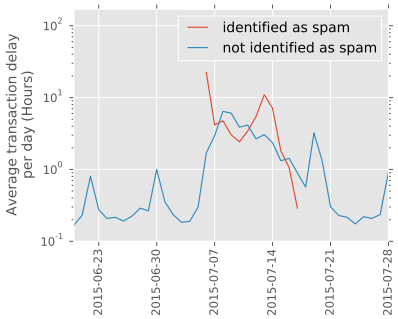

Fig. 4. Average transaction delay between when a transaction appears in the Mempool and when it is committed to the blockchain.

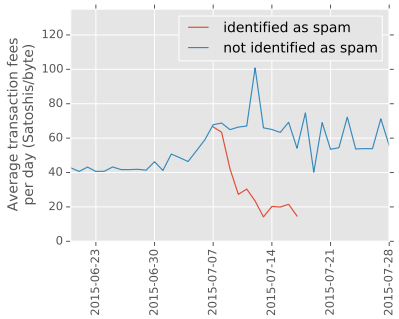

Fig. 5. Average transaction fees per transaction per day. Note that the fees are normalized against the size of each transaction.

6 Discussion

The spam campaign happened when the Bitcoin network is divided regarding a critical component of the protocol: the block size limit of 1 MB. The result was a recent fork between two camps: some want to raise the limit while others refuse to alter the rules set by Satoshi Nakamoto (Bitcoin's creator). We can speculate that the spammer was motivated to launch a DoS attack to demonstrate the fragility of Bitcoin's resilience if the block size limit is not raised. This is supported by an earlier spam campaign, where an online Bitcoin wallet service claimed responsibility under the pretext of "stress testing".

With regards to the methodology proposed in this paper, we do not suggest that this model can be used to prevent Bitcoin spam completely, nor should it be used as such. It was used to measure and analyze spam after the fact, without the spammers being aware that they are being measured. Spammers can learn from this paper what heuristics and features we used, to alter their motifs and adapt accordingly.

Although we used dust checks to validate our results, this is not a fool-proof measurement to accurately validate spam in Bitcoin. It is trivial to create a transaction that does not generate any dust outputs. However, if a transaction does not create dust, then the clustering algorithm matches that transaction to

a cluster that highlights other features, particularly differences in ratio. We use dust validation when there is a possibility a cluster contains normal transactions. Since we defined unusual fan-out ratios to constitute spam transactions (and they are gathered in distinct clusters), along with our conservative measurement of dust, we believe that the results we had shown earlier provide a good estimate of the July spam campaign.

It is important to note that the July spam campaign on Bitcoin would be infeasible on altcoins since some deploy a different model for transaction fees. For example, Litecoin charges the `mintxfee` for *each* small output. Bitcoin can adopt a similar model, or a dynamic model for fees, possibly using clustering results to observe spam patterns, and change `mintxfee` accordingly.

7 Related Work

In their recent Bitcoin SoK paper [2], Bonneau et al. highlight open research questions and discuss issues with Bitcoin regarding stability and scalability, ranging from Bitcoin forks and network analysis to incentivizing correct behavior and adding resilience with proposed changes. The authors highlight *penny flooding*, as discussed in [8], which is related to our discussion of dust transactions. In the Appendix of the extended version of their SoK paper, the authors discuss in more details Bitcoin's stability and transaction validity. The extended version includes a discussion outlining options to overcome the drawbacks of maintaining the entire UTXO to process new transactions: using a *statefile*, updated incrementally with new data, it is possible to more efficiently retrieve transactions for verification using a transaction's hash in $O(log\ M)$, where M is the number of unspent transactions. It also possible to further minimize the data structure required to validate transactions using hash-based authenticated data structures as proposed in [5].

Becker et al. [1] discuss the possibility of denying service to the Bitcoin network using a virtual protest: protestors join forces to collectively execute a DoS attack by overwhelming the network and depleting precious block space (these transactions are much larger than normal Bitcoin transactions). If a protest is ongoing, this can frustrate non-protestors and decrease faith in the resilience of Bitcoin to process transactions in a timely manner (Bitcoin's main features include processing payments in minutes, as well as low transaction fees). This virtual protest attack was labeled 'Occupy Bitcoin' by Kroll et al. [4].

Our clustering approach is different than previous work aiming to find patterns in Bitcoin: we strip away identifying information (such as *txid*, addresses, etc.), and cluster transaction features in order to determine patterns, rather than linking transactions together to de-anonymize users. For example, in [6], researchers cluster transactions based on determining shared authority properties, while using similar transaction features used in this research.

Other empirical research determining detrimental affects on the Bitcoin network measure the lifetime of Bitcoin exchanges [7], through analyzing daily transaction volumes and how exchange breaches affect survival time. Running a

profitable exchange logically results in a more lucrative target, hence a breach is more likely which leads to the eventual shutdown of an exchange. Another approach is to parse online forums to obtain data indicators of possible attacks on the network, as was done in [11].

8 Conclusion

We have presented an empirical study of a spam based "stress test" DoS attack against Bitcoin. Using our clustering based approach we find that 385,256 (23.41%) out of 1,645,667 total Bitcoin transactions were spam during a 10 day period at the peak of the spam campaign. We also show that this attack had a negative impact on non-spam transactions, increasing average fees by 51% (from 45 to 68 Satoshis/byte) and processing delay by 7 times (from 0.33 to 2.67 hours). This shows that an adversary who is willing to expand modest amounts of bitcoin (at least $49,000 USD), to pay higher fees, can DoS Bitcoin. Follow up DoS attacks against Bitcoin have used other methods, such as "money drops", and transaction malleability to degrade the operation of Bitcoin. We point out that changes to Bitcoin's minimum fees could mitigate some of the spam motifs we witnessed. Our results show that exploration into Bitcoin transaction spam filtering techniques, and other Bitcoin DoS mitigation approaches, merit further investigation.

Acknowledgements. This work was supported by US National Science Foundation grant CNS-1619620.

References

1. J. Becker, D. Breuker, T. Heide, J. Holler, H. P. Rauer, and R. Böhme. Can we afford integrity by proof-of-work? Scenarios inspired by the Bitcoin currency. In *The Economics of Information Security and Privacy*, pages 135–156. Springer, 2013.
2. J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten. SoK: Research perspectives and challenges for Bitcoin and cryptocurrencies. *IEEE Symposium on Security and Privacy*, 2015.
3. C. Decker and R. Wattenhofer. Information propagation in the Bitcoin network. In *IEEE Thirteenth International Conference on Peer-to-Peer Computing (P2P)*, pages 1–10, 2013.
4. J. A. Kroll, I. C. Davey, and E. W. Felten. The economics of Bitcoin mining, or Bitcoin in the presence of adversaries. In *Proceedings of WEIS*, volume 2013, 2013.
5. G. Maxwell. Merkle tree of open transactions for lite mode? bitcointalk.org, 2011.
6. S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage. A fistful of Bitcoins: characterizing payments among men with no names. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 127–140. ACM, 2013.
7. T. Moore and N. Christin. Beware the middleman: Empirical analysis of Bitcoin-exchange risk. In *Financial Cryptography and Data Security*, pages 25–33. Springer, 2013.

8. M. Möser and R. Böhme. Trends, tips, tolls: A longitudinal study of Bitcoin transaction fees. In *2nd Workshop on Bitcoin Research, affiliated with the 19th International Conference on Financial Cryptography and Data Security, Puerto Rico*, 2015.
9. S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Consulted*, 1(2012):28, 2008.
10. M. Trillo. Stress Test Prepares VisaNet for the Most Wonderful Time of the Year. `http://www.visa.com/blogarchives/us/2013/10/10/stress-test-prepares-visanet-for-the-most-wonderful-time-of-the-year/index.html`, 2013.
11. M. Vasek, M. Thornton, and T. Moore. Empirical analysis of denial-of-service attacks in the Bitcoin ecosystem. In *Financial Cryptography and Data Security*, pages 57–71. Springer, 2014.