Countering Race Attacks for Zero-Confirmation Transactions in the Bitcoin Network

Christopher J. Baumler

***Abstract*—The Bitcoin network is used to conduct payment transactions which are validated using a proof-of-work (PoW) system. This system is designed to prevent double spending of Bitcoins (BTCs), provided the majority of the computation power performing the PoW is honest. However, the system is not well suited for fast payments such as those required for vending machines or Automated Teller Machines (ATMs). In this paper the problem of race attacks on zero-confirmation transactions is defined, and several solutions from the literature are analyzed. Finally, we offer a solution that improves upon existing methods.**

***Index Terms*—Bitcoin, Race attack, Double spending**

I. Introduction

Bitcoin is a distributed payment system and currency that uses a proof-of-work (PoW) system to acknowledge and validate transactions communicated over the network. Transactions contain information such as the number of Bitcoins (BTCs) being transferred, the original owner of the BTCs, and the new owner of the BTCs. Participants in the transaction are identified by means of Bitcoin addresses which map to unique public/private key pairs. Ownership of a BTC equates to knowledge of the private key linked to the Bitcoin address to which the BTC has been transferred [1]. The PoW system is designed such that it requires an average time of 10 minutes to verify a block of transactions and add it to the public record called the blockchain [5]. The length of time required to verify a transaction does not lend itself to systems that require fast transactions such as Automated Teller Machines (ATMs) or retailers. However, failure to wait for transaction verification leaves the BTC recipient vulnerable to a problem called double spending where the BTC is spent more than once [6]. This paper describes the basic operation of the Bitcoin network and defines the problem of needing an effective verification method for fast transactions.

II. The Bitcoin network

The Bitcoin disturbed payment system is made up of a network of peers running a Bitcoin client. The peers issue, communicate, approve, and acknowledge transactions. A transaction is created by digitally signing a hash of the previous transaction involving the BTC along with the public key of the new owner and appending the signature to the coin. Thus, the ownership of the BTC can be verified by checking the signatures associated with it [1].

Once a transaction is created, it is broadcasted to peers in the network who verify that the transaction is properly formed, and that the transaction details do not conflict with the public transaction history stored in the blockchain. As transactions are accumulated, the peers incorporate them into blocks and attempt to solve a PoW. The PoW involves adding a nonce to the set of transactions and computing a signature using a hash function. If the signature is within a specified range, the PoW is solved, and the peer broadcasts the block to the network. If the signature is not within the specified range, the peer continues to attempt a solution of the PoW by incrementing the nonce and computing a new signature. When a peer submits a block that is successfully accepted into the blockchain, it is rewarded with newly minted BTCs [1].

When a block is broadcasted on the network, the peers verify that the block hash is a valid solution to the PoW and that the transactions within the block do not conflict with the public transaction history. If a block is successfully verified, it is added to the peer's blockchain, and the peer begins computing the next block using the hash of the verified block. If two valid blocks are broadcasted to the network in close time proximity, peers accept whichever block is received first, creating a branch in the blockchain. Ultimately, the branch containing the block that is accepted by the most peers becomes the main branch, and the other branch is abandoned. Blocks in abandoned branches are referred to as orphans [7].

To double spend BTCs, an adversary needs to alter the blockchain. This means issuing a second transaction involving the previously spent BTCs, thereby creating a branch in the blockchain. This branch must become the main branch by becoming longer than the branch containing the original transaction. However, to achieve this, each subsequent block that has been appended to the block containing the original transaction must have its PoW redone and be appended to the new branch. Accomplishing this becomes increasingly difficult as new blocks are added to the blockchain. Thus, ideally, a payee should wait until a transaction has been verified and added to the blockchain before offering a service to a payer. However, this is not acceptable for some situations which require fast payments, leaving the payee vulnerable to double spending. This double spending threat poses a serious roadblock to the adoption of Bitcoin as a method for carrying out fast transactions and should be addressed.

III. Types of Double Spending Attacks

Several different categories of double spending attacks have been defined. Among these are race attacks, Finney attacks, Vector76 attacks, brute force attacks, and 51% attacks [6]. In this section we briefly describe each of these, and in the next section, we focus on race attacks specifically, as they are the most relevant to zero-confirmation transactions involving amounts in the low to middle price range.

A. Race Attacks

The least resource intensive form of double spending attack is known as the race attack. The basic premise of this attack is to communicate one transaction to a vendor while simultaneously communicating a second transaction involving the same BTC to the rest of the network, ensuring that the alternate transaction will be included in the blockchain, while tricking the vendor into providing the desired service [6]. This attack has been shown to be feasible in [3] and [4], and it is the focus of this paper.

B. Finney Attacks

The Finney attack is similar to the race attack but is more expensive in that it requires the attacker to enlist the participation of a miner [6]. Here the dishonest miner works on a block containing a transaction involving the attacker's BTC. When the PoW is solved for the block, the attacker quickly conducts another transaction spending the same BTC. Then the block is submitted, and the second transaction is invalidated. This form of attack is costly since significant hardware is necessary to guarantee that enough blocks will be generated to make the attacks worthwhile. Additionally, the attacker runs the risk that another node will submit a valid block while the second transaction is being conducted, costing the miner the reward for successfully submitting a block. Due to the significant cost involved, this attack is not likely to be performed against vendors offering goods and services in the low to middle price range [8].

C. Vector76 Attacks

The Vector76 attack builds upon the race and Finney attacks. Here the attacker mines a block containing a transaction with a valid deposit to a vendor. The attacker waits until another block has been mined by a different node and then transmits the block to the target node. The target node will accept the node with one confirmation. However, nodes that receive the other block first will accept that block into their copies of the blockchain. Eventually, if the target node whose blockchain contains the attacker's block loses the race to lengthen its chain, the block will become an orphan and be invalidated. Like the Finney attack, this attack requires significant resources. In particular the attacker must sacrifice the reward for successfully generating a block [6].

D. Brute Force Attacks

To conduct a brute force attack, the attacker must control a significant portion of the total network hashrate. Essentially, the attacker must create a fork in the blockchain and cause the fork to grow longer than the original chain. This becomes much less likely as the number of confirmations increases [6].

E. 51% Attacks

The 51% attack is a specific instance of a brute force attack where the attacker controls more than half of the network hashrate. In this case the attacker's branch of the blockchain will become longer than the honest branch with 100% probability. Thus, no number of confirmations will protect against it [6].

IV. Performing a Successful Race Attack

In [3] a model is proposed for successfully performing a race attack. Several assumptions are made in the proposed model. For example the model assumes that the attacker knows the Bitcoin and IP addresses of the vendor. Additionally, the attacker only has the resources to control a few peers in the network without a significant portion of the hashrate and does not have access to the vendor's private keys or machine. Finally, the other peers in the network are assumed to be honest and following the Bitcoin protocol.

In performing the race attack, the attacker creates two transactions, TR_V and TR_A. TR_V transfers ownership of some BTC to the vendor and is sent to the vendor's node which propagates the transaction to its neighbors. TR_A transfers ownership of those same BTC to another address, possibly owned by the attacker, and is sent to a helper node which propagates the transaction to its neighbors. For the attack to succeed, three conditions must be met.

A. Condition 1

The first condition is that TR_V must be the transaction accepted by the vendor's wallet. This means that TR_V must reach the vendor prior to TR_A, because when nodes receive two conflicting transactions, the first one is accepted as valid, and the second is discarded. To satisfy this condition, the attacker should attempt to establish a direct connection to the vendor's node or to a node close to the vendor's node whereby TR_V may be sent. If the vendor's node is configured to accept incoming connections, this task is fairly easily accomplished. Given the dynamic nature of the Bitcoin network, the number of peers connected to a node changes frequently, and even if the vendor's node is using the maximum number of connections, the number is likely to drop below the maximum at various points in time. Likewise, the attacker should send TR_A to a helper node which is not connected to the vendor's node or ideally to any of the node's neighbors. This ensures that the vendor will receive TR_V first, since there will be more hops in the path TR_A must follow to reach the vendor [3].

B. Condition 2

The second condition states that TR_A must be confirmed in the blockchain. This is necessary to ensure that the attacker gets the BTC back. To satisfy this condition, the node that successfully performs the PoW for the next block must have received TR_A before receiving TR_V. The likelihood of this happening depends on the percentage of nodes in the network which accept TR_A as the valid transaction. Note that since honest nodes always propagate the transaction that they perceive to be valid and discard invalid transactions, both TR_V and TR_A are being spread across the network. However, not all nodes actually see both transactions. In [3] the probability that the second condition is satisfied is found to be determined as follows:

$$P_s = Prob(t_{ga} < t_{gv}) + \frac{1}{2} Prob(t_{ga} = t_{gv})$$

where t_{ga} is the time required by nodes that have accepted TR_A to generate a new block, and t_{gv} is the time required by nodes that have accepted TR_V to generate a new block.

Further analysis done in [3] has shown that the attacker can maximize P_s by maximizing the number of nodes that receive TR_A first. One way to achieve this is to send TR_A before TR_V. This gives TR_A a head start in propagating through the network, and as long as the vendor receives TR_V first, the first condition is still satisfied. Alternatively, the attacker could employ multiple helper nodes to spread TR_A through the network at a faster rate.

C. *Condition 3*

The last condition that must be met to successfully perform a race attack is to ensure that the time necessary for receiving service from the vendor is smaller than the time necessary for the vendor to detect the attack. The assumption is made here that since the Bitcoin user is anonymous, the vendor cannot take any recourse after the fact. This condition is easily met if the vendor is using a Bitcoin client that does not alert the user when a transaction is received whose inputs have already been spent [3].

V. Overview of Proposed Solutions

Analysis done by [3] has shown that the conditions necessary to perform a race attack can feasibly be achieved with high probability and few resources. Thus, an efficient technique to prevent these attacks is highly desirable for widespread adoption of Bitcoin in applications that must accept zero-confirmation transactions. Several different solutions have been proposed which have various shortcomings. These solutions are discussed in this section, and a hybrid solution which attempts to build upon these solutions is proposed in the following section.

A. *Listening Period*

One possible answer to the race attack is to implement a “listening period” where the vendor waits for a certain number of seconds before delivering the product or service to the customer. During this time the vendor’s node listens for a second invalid transaction which would alert the vendor to the attack [9]. This approach appears viable, because the average time for a transaction to reach all of the nodes in the Bitcoin network is on the order of a few seconds. However, while this method has been shown to reduce the success rate of race attacks, it can be circumvented [3].

To successfully conduct a race attack against a vendor who employs a listening period approach, an attacker must delay the transmission of TR_A such that TR_A is received after the listening period has expired. The attacker must also ensure that TR_A is present in the generated block. To increase the probability of success, the attacker can employ additional helper nodes to spread TR_A more quickly. Experiments performed by [3] have shown that when delaying the transmission of TR_A by various times and utilizing three or four helper nodes, an attacker can achieve a wide range of success rates. Additionally, a special range of parameters exist such that the attacker can guarantee that the vendor will not ever detect the attack regardless of the listening period. This is possible, because if all of the neighbors of the vendor’s node receive TR_V first, they discard TR_A rather than forwarding it to the vendor’s node. Furthermore, the probability of the attack succeeding under these conditions is within the range of 5% to 13.33%.

B. *Observers*

Another solution proposed by [9] requires the vendor to place additional nodes into the network which relay the transactions they receive back to the vendor’s node. This allows the vendor to become aware of TR_A more quickly. However, it adds additional expense and complexity to the system that must be implemented and maintained by the vendor.

As with the listening period technique, this solution was evaluated by [3] and found to be only partially successful. In the experiments performed, five observers were used, and the percentage of attacks observed ranged from 18% to 91%.

C. *Forwarding Double Spending Attempts*

The technique proposed by [3] to defeat the race attack involves making a modification to the Bitcoin protocol. Currently, nodes are required to drop all invalid transactions including those that attempt to double spend BTC. The proposed change would require nodes to forward all double spend transactions to their neighbors. Specifically, nodes would verify new transactions against the transactions in blockchain and in their memory pool. If a transaction is valid, it would be added to the memory pool and forwarded to the node’s neighbors. If the transaction includes coins that have been spent already, the transaction would still be forwarded to the node’s neighbors but would not be saved in the memory pool. Notably, this solution does not do anything to change the rate at which TR_V and TR_A are propagated.

Implementing this change in the Bitcoin protocol would ensure that all nodes in the network would eventually receive both TR_V and TR_A. Thus, guaranteeing that the vendor could detect a race attack by listening for a period of time on the order of a few seconds. Experimentation in [3] determined this time to be approximately 3.354 seconds. Additional experimentation has shown that the technique has a detection rate of 100% and a false negative rate of 0%.

The major drawback of the solution proposed by [3] is the added communication burden imposed upon the Bitcoin network. The additional load may be tolerable when the network is composed of honest nodes. However, the modified

Bitcoin protocol would leave the network open to a Denial of Service (DOS) attack. An attacker with a small number of BTC could feasibly generate a large number of invalid transactions that double spend those BTC, possibly using a botnet. Each of these invalid transactions would be propagated to by the Bitcoin network to every node in the network, significantly multiplying the effect of the attack. The threat of a DOS attack is particularly relevant given that such attacks have been conducted in the past [10].

D. Verifiable Code Execution

The final technique for countering race attacks, proposed by [2], differs from the previously described solutions in that it does not seek to detect attempts at double spending but rather tries to establish mutual trust between the vendor and the customer. This is accomplished through the use of verifiable code execution. Specifically, a modified version of the Pioneer model is proposed.

The Pioneer model of verifiable code execution uses a challenge-response protocol between a trusted entity known as a dispatcher and an untrusted entity called the untrusted platform. Essentially, the protocol offers proof that an executable segment of code is invoked without having been modified within an uncompromised execution environment. The original Pioneer only provides one way trust. That is, the vendor can trust that the customer has not tampered with the Bitcoin client, but since the executable client code comes from the vendor, the customer cannot trust that the vendor has left the client code unmodified. To rectify this [2] proposes a modified version of the Pioneer model where the customer's node is first able to verify that the vendor has not modified the Bitcoin client, and then the vendor is able to do likewise. In this fashion mutual trust is established between the two nodes.

The verifiable code execution solution is proposed as an optional layer to be built on top of the current PoW based system. It offers two main advantages. First, it does not interfere with the existing Bitcoin network or necessitate updates to the network. Second, it requires much less time to execute this technique than any of the previously discussed solutions. Simulations performed by [2] have shown that the total time required to confirm a transaction is approximately 186.25ms.

The benefits of the verifiable code execution technique make it an attractive solution. However, the solution has limitations stemming from the underlying assumption that the attacker must tamper with the client that is issuing the transaction in order to conduct an attack. As we have mentioned previously, a condition of the race attack is that the attacker must send TR_V to the vendor's node. This could presumably be done without modifying the standard Bitcoin client, thus allowing verifiable code execution. However, the attacker also employs a helper node which propagates TR_A. Since this node is also controlled by the attacker, a slight modification to the original attack can be undertaken. The helper node may be given access to the private keys of the BTC provided as inputs to TR_V, allowing the helper to create TR_A in parallel to the creation of TR_V. Through the use of a timing mechanism, the helper could propagate TR_A into the network at the desired instant. The helper node does not need to communicate with the vendor node at all. So verifiable code execution fails to prevent a race attack in this scenario.

VI. A Hybrid Approach

Each of the solutions proposed in the literature has strengths and weaknesses when considered individually. However, we propose that a hybrid approach incorporating elements from several techniques may be a more effective defense against race attacks. By combining various techniques, we arrive at the following hybrid solution to be employed by the vendor's node:

1) Refuse incoming connections from nodes.
2) Establish as many connections as possible with well-known nodes.
3) Establish one or more observer nodes.
4) Enforce a listening period.
5) Modify the Bitcoin client such that transactions sending money to the vendor, such as TR_V, are not propagated.
6) Modify the Bitcoin client to alert the vendor when an attack is detected.

Step (1) is suggested by [11] as a best practice for configuring vendor nodes. By refusing incoming connections, the vendor makes the race attack more difficult to perform. This is because the attacker is no longer able to connect directly to the vendor to ensure that TR_V is delivered quickly without any hops. This alone is not a viable defense, because in theory the attacker could locate a neighbor to the vendor's node to obtain similar results to connecting directly to the vendor's node.

In step (2) connecting to as many nodes as possible is desirable. Experiments in [3] suggest that the more connections the vendor's node has, the lower the probability of success the attacker has. In addition to the number of connections, the quality of the connections should be considered as well. Connecting to well-known mining pools is suggested by [11]. This may not be feasible, as mining pools can only provide a limited number of connections. However, in the future, companies may foreseeably exist that provide well-known nodes with connections to mining pools as a service.

Steps (3) and (4) are incorporated from proposed solutions discussed previously in this paper. Step (4) is essential for detecting TR_V, and step (3) is included to increase the detection rate. The number of observers may be determined based on a comparison of the desired degree of protection required versus the acceptable cost of implementation.

Step (5) has been suggested by [11]. By not propagating TR_V, the vendor's node effectively allows all of its neighbors to become observers. This should reduce the probability that the vendor's node does not ever receive TR_A. Alternately, the vendor's node might opt to send TR_V to some but not all of its neighbors.

The last step is necessary to make the vendor aware of the attack. Once alerted, the vendor may refuse to offer goods or services to the attacker. The vendor may desire to take additional steps such as refusing to refund spent BTC in the event that a race attack fails. By implementing this last step along with the others, a more comprehensive defense against race attacks may be enforced.

VII. Testing The Hybrid Approach

To evaluate the proposed hybrid approach, a test environment was constructed, and a series of tests were conducted, varying different system parameters.

A. *Test Environment*

The test environment consisted of three components: an attacker, a vendor, and an observer. Each of these components was implemented on a different platform and was connected to the Bitcoin peer-to-peer network using the Bitcoin-QT Bitcoin client. Fig. 1 shows how the test environment was arranged.

1) *Attacker Component*

The attacker component was implemented on a computer running 64-bit Windows 7. The system consisted of two nodes and a controller program. Each node utilized version 0.9.1.0-g026a939-beta of the Bitcoin Core software and version 5.2.0 of the QT software. The system was set up such that both nodes shared the same wallet file but used different copies of the blockchain. Thus, each had access to the same inputs for spending and tracked updates to the blockchain separately.

A controller program was written in Python to send commands to the attacker nodes. The program communicated with the nodes using remote procedure calls and the API provided by the Bitcoin client. When executed the program queried the nodes for unspent Bitcoin inputs using the "listunspent" API and proceeded to create two raw transactions that used the same unspent input by calling the "createrawtransaction" API. One transaction, TR_V, was assigned a destination address belonging to the vendor. The other transaction, TR_A was assigned an address belonging to the attacker. Next, the transactions were signed using the "signrawtransaction" API and sent using the "sendrawtransaction" API. Each transaction was sent to a different node, and transaction TR_A was sent after TR_V with a configurable delay. Finally, the controller program displayed the transaction signatures for easy identification later.

The two attacker nodes were configured such that they were not directly connected to each other on the network. One node was designated to send TR_V and was only allowed to connect to a single random peer on the network. The other node was designated to send TR_A and was allowed to connect to 8 random peers.

During setup of the attacker component, some issues were encountered and overcome. For example, to run both attack nodes on the same PC, it was necessary to configure one of the nodes not to listen for incoming traffic on the designated port for Bitcoin network traffic, port 8333. This modification did not affect the results of the tests, as it only limited the ability of other peers to connect to the attacker node and did not limit the attacker nodes ability to initiate communication with peers. Additionally, it was discovered that Bitcoin transactions

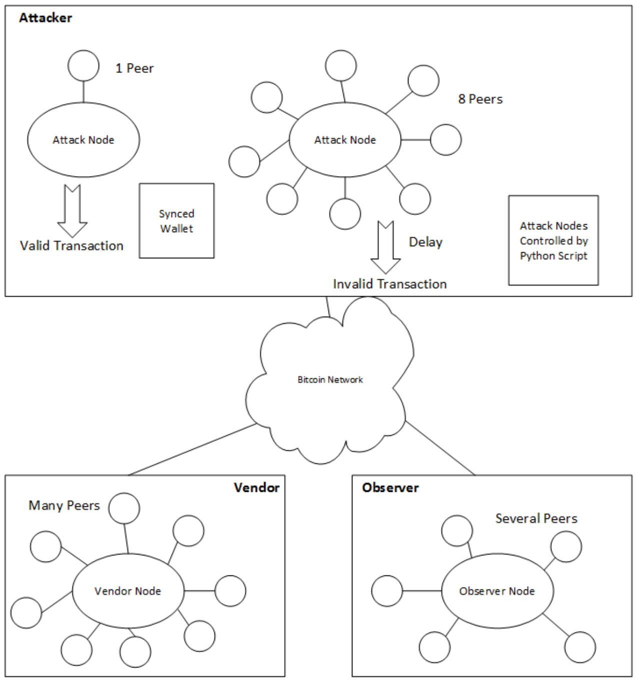


Fig. 1. Test environment for testing the hybrid approach to detecting double spending attacks. Note that each component (attacker, vendor, observer) is implemented on a separate platform. They are connected through the Bitcoin network.

containing small amounts must include a transaction fee of 0.0001 Bitcoin to be added to the blockchain in a timely manner. Without this fee, many miners will not accept the transaction, and the likelihood of the transaction being included in a mined block is much reduced. During testing we observed a waiting period of as much as 8 hours for transactions of 0.1 Bitcoin when no fees were allocated. To add a fee to a raw transaction, a portion of the input amount was not allocated to an output. This automatically assigns that amount as a transaction fee.

2) *Vendor Component*

The vendor component was implemented on a virtual machine running 64-bit Ubuntu Linux. The system consisted of a single node utilizing version 0.9.1.0-g026a939-beta of the Bitcoin Core software and version 5.2.0 of the QT software. The Bitcoin software source code was modified and recompiled to implement the hybrid approach to detecting double spending.

To implement the hybrid approach, the functionality for re-transmitting received transactions was removed. In general only refusing to relay transactions involving the vendor is sufficient. However, for ease of implementation, all relaying of transmissions was disabled in the test environment. Additionally, functionality was added to alert the vendor when a double spend occurred. This consisted of messages printed to the vendor node's console. Finally, the hard-coded maximum for allowed outgoing peer connections was changed from 8 to 1000. This value is presumably set low by default to prevent nodes from unnecessarily consuming all available connections to peers.

In addition to the code changes made to the vendor node, a set of configuration parameters were specified to implement the hybrid approach. These included disabling incoming connections from peers and setting the maximum total number of connections allowed for the node to 1000. Additionally, a configuration file was created containing a list of several hundred peer addresses for the node to use in establishing outgoing connections. Lastly, the node was configured to add debug data pertaining to its transaction memory pool to a log file. This included data such as when valid transactions were received and added to the pool, when invalid transactions were received along with why they were invalid, and when transactions resulted in Bitcoins being added to the vendor's wallet.

3) *Observer Component*

The observer component was implemented on a computer running 64-bit Windows 7. The system consisted of a single node utilizing version 0.9.1.0-g026a939-beta of the Bitcoin Core software and version 5.2.0 of the QT software. The Bitcoin software on this node was unmodified. The node was configured to allow both incoming and outgoing connections in order to establish as many connections as possible to random peers. However, due to the implementation of the Bitcoin software, the node was limited to 8 outgoing connections. This was not changed for the test, because one of the disadvantages of using observers is complexity and cost, and using unmodified Bitcoin software for the observer node reduces both cost and complexity. In addition to connection configuration, the observer node was configured to display debug data in a log file similar to the vendor node.

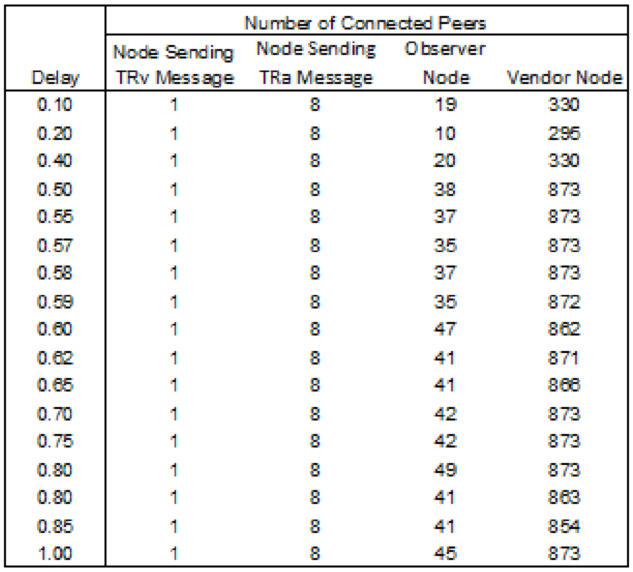
B. *Tests*

A range of tests were conducted primarily varying the delay between the initial propagation of TR_V and TR_A. A test was initiated by configuring a delay period and executing the Python controller program in the Attacker component. This caused TR_V and TR_A to be transmitted to the Bitcoin peer-to-peer network.

After a test had been initiated, we examined the log files produced by the vendor and observer nodes. To enforce a reasonable listening period, only data logged within 10 seconds of initial propagation of TR_V and TR_A was considered valid. We searched for messages indicating receipt of TR_V and TR_A as well as messages indicating if a double spend had been detected. We also used the "getpeerconnections" command to determine the number of peers connected to the vendor node and the number of peers connected to the observer node.

In addition to checking log files, we used the graphical interface provided by the Bitcoin software to observe when a transaction was received by a node and when it was confirmed in the blockchain. The interface was also used to help verify when a double spend occurred. This was indicated by a transaction record marked as conflicted.

The test procedure described previously was executed a total of 17 times. The results are shown in Fig. 2, Fig. 3, and Fig 4. Fig. 2 shows the number of connected peers for each component. As previously noted, the number of peers connected to the node transmitting TR_V was always 1. This was done to prevent TR_V from propagating too quickly.



Delay	Number of Connected Peers			
	Node Sending TRv Message	Node Sending TRa Message	Observer Node	Vendor Node
0.10	1	8	19	330
0.20	1	8	10	295
0.40	1	8	20	330
0.50	1	8	38	873
0.55	1	8	37	873
0.57	1	8	35	873
0.58	1	8	37	873
0.59	1	8	35	872
0.60	1	8	47	862
0.62	1	8	41	871
0.65	1	8	41	868
0.70	1	8	42	873
0.75	1	8	42	873
0.80	1	8	49	873
0.80	1	8	41	863
0.85	1	8	41	854
1.00	1	8	45	873

Fig. 2. Number of peers connected to each node in the test environment. Note that the number of connections allowed for the attacker nodes was fixed, while peers connected to the observer and vendor nodes were limited but allowed to vary based on available connections.

Likewise, the node transmitting TR_A was always connected to 8 peers to ensure TR_A was propagated more quickly than TR_V. Since the observer node was limited to only 8 outgoing connections, the number of connected peers varied as other nodes in the network connected and disconnected. The number of connected peers tended to increase initially, eventually reaching a steady state with an average of about 40 to 45 connected peers. The number of peers connected to the vendor node was much larger than any of the other nodes. This was done by design to satisfy part (2) of the hybrid approach. The eventual steady state number of peers was around 870. However, several hours were required to reach this level of connectivity.

Fig. 3 shows whether nodes observed TR_V or TR_A during a test run. Results are recorded for both the vendor node and the observer node. The last column of the table shows which transaction was ultimately accepted into the blockchain.

Fig. 4 shows whether each of the three conditions necessary for conducting a successful race attack was met for a given test run. This data was used to evaluate the effectiveness of the hybrid approach to detecting double spending attacks.

VIII. Evaluating The Hybrid Approach

The results of each test were evaluated against the three conditions required to successfully conduct a race attack. Results were then analyzed to determine the probability of successfully executing a race attack and were compared against other proposed techniques.

A number of observations may be made by viewing the test results depicted in Fig. 3 and Fig. 4. First, we note that in general, the probability that TR_V is detected by the vendor increases as the delay between transmission of TR_V and TR_A increases. This aligns with our theoretical knowledge of message propagation in the Bitcoin peer-to-peer network. We also see that when the attacker is not allowed to connect

Delay	TRv Observed		TRa Observed		Accepted Into
	By Vendor	By Observer	By Vendor	By Observer	Blockchain
0.10	no	no	yes	yes	TRa
0.20	no	no	yes	yes	TRa
0.40	yes	no	yes	yes	TRa
0.50	no	no	yes	yes	TRa
0.55	no	no	yes	yes	TRa
0.57	yes	yes	no	no	TRv
0.58	no	yes	yes	no	TRv
0.59	yes	yes	no	no	TRv
0.60	no	no	yes	yes	TRa
0.62	yes	yes	no	no	TRv
0.65	yes	yes	no	no	TRv
0.70	yes	yes	no	no	TRv
0.75	no	no	yes	yes	TRa
0.80	yes	yes	no	no	TRa
0.80	yes	yes	no	no	TRv
0.85	yes	yes	no	no	TRv
1.00	yes	yes	no	no	TRv

Fig. 3. Observation of TR_V and TR_A for the vendor and observer nodes along with a record of which transaction was ultimately accepted into the blockchain.

Delay	Condition 1	Condition 2	Condition 3
0.10	FALSE	TRUE	TRUE
0.20	FALSE	TRUE	TRUE
0.40	TRUE	TRUE	FALSE
0.50	FALSE	TRUE	TRUE
0.55	FALSE	TRUE	TRUE
0.57	TRUE	FALSE	TRUE
0.58	FALSE	FALSE	FALSE
0.59	TRUE	FALSE	TRUE
0.60	FALSE	TRUE	TRUE
0.62	TRUE	FALSE	TRUE
0.65	TRUE	FALSE	TRUE
0.70	TRUE	FALSE	TRUE
0.75	FALSE	TRUE	TRUE
0.80	TRUE	TRUE	TRUE
0.80	TRUE	FALSE	TRUE
0.85	TRUE	FALSE	TRUE
1.00	TRUE	FALSE	TRUE

Fig. 4. This table depicts whether each of the three conditions necessary for conducting a successful race attack are met for a given test run. Note that only one test run satisfied all three conditions.

directly to the vendor node, the first condition becomes much harder to meet. Of the 17 tests executed as part of this experiment, the vendor observed TR_V in only 10 cases. This means that 7 of the attacks failed simply because the vendor never received the valid transaction, suggesting that implementing step (1) of the hybrid approach does indeed have a beneficial effect. We also note that for our test environment, the range of delay values between 0.5 and 0.8 seems to be ideal for conducting race attacks. Within this range, we see both TR_V and TR_A regularly observed by the vendor, and we see both transactions being accepted into the blockchain.

In general, we see an inverse relationship between observation of TR_V and TR_A. Both were observed simultaneously by the vendor in only one test instance. This result makes sense given that nodes do not propagate transactions they deem invalid. However, we might expect to observe both transactions in more cases given the large number of peers connected to the vendor node. As such the effectiveness of step (2) of the hybrid approach seems to be limited. However, it is possible that, given a larger sample size, instances where both TR_V and TR_A are detected would increase. Experiments performed in [3] certainly suggest as much.

Fig 3. also shows data regarding the single observer node in the system which was implemented as part of step (3) of the hybrid approach. We note that the observer, which had fewer connected peers than the vendor, never received both TR_V and TR_A in the same test instance. However, for the observer to be useful, it only needs to observe the opposite transaction as the vendor node and send that information to the vendor in a timely fashion. This was the case in the test instance with a delay of 0.58 where the vendor received only TR_A and the observer received TR_V. Thus, the observer would have allowed the vendor to detect the attempted double spend. Although, in this case, the detection was of limited utility given that the vendor would not have dispensed any goods anyways. However, more usefulness may be found in the inverse case where the vendor receives only TR_V, and observer receives TR_A. In this second case, if TR_V is accepted by the blockchain, as a penalty for double spending, the vendor may opt to both refrain from dispensing goods and refuse a refund of the purchasers Bitcoins.

Discussions of whether a vendor should dispense goods must include a timeframe for providing said products. In our tests, step (4) of the hybrid approach was implemented with a minimum listening period of 10 seconds. This listening period appeared to be adequate, as no additional beneficial information was observed after that period.

Analyzing whether step (5) of the hybrid approach was beneficial apart from steps (1) and (2) is difficult. We can analyze the expected effect of this step, and we see that it is vitally important in an environment where the attacker is able to connect directly to the vendor node or to a node adjacent to the vendor node. However, this step may provide less benefit in an environment such as ours where the attacker is limited to connections to random peers. Ultimately, this step should be included if only because it may offer some benefit while causing no negative effects.

The last column of the table depicted in Fig. 3 shows which transaction was accepted into the blockchain for each test instance. As we have previously noted, for a race attack to succeed, TR_A must be accepted into the blockchain. This was the case in 8 of the 17 tests. This even split between TR_V and TR_A seems reasonable given the spread of delay times. We observe that it becomes necessary to strive for a fairly even propagation of TR_V and TR_A through the network to conduct a race attack when one cannot determine when the vendor will receive TR_V as was the case in [3]. Further, the advantage gained by delaying the transmission of TR_A and using more nodes to propagate TR_A is nullified.

Several observations can be made about the results in Fig. 4. For example, we observe that two race attacks were detected, one with delay 0.4, and one with delay 0.58. The vendor was notified of these attacks thanks to the implementation of step (6) of the hybrid approach. On the other hand, one race attack succeeded. All three conditions were met for the test instance with delay 0.8. This means the detection rate was 11.76%, and the attack success rate was 5.88%. The detection rate in particular seems low. This may be linked to the high success rate of condition 3. The success rate of condition 3 was high because several of the test cases saw one or the other of the transactions propagating through the network much more quickly than the other, resulting in a scenario where one message was isolated from the majority of the network. However, these scenarios are not ideal for a successful race attack. This conclusion is supported by the low attack success rate.

The attack success rate of 5.88% was lower than rates reported in [3] for most other detection methods. However, the rate reported for the "Forwarding Double Spending Attempts" method was 0%. Yet ultimately, the hybrid approach may be superior to this method, since the "Forwarding Double Spending Attempts" method requires a change to all peers operating on the Bitcoin network and leaves the network open to denial of service attacks.

IX. Further Research

The results of this paper may be expanded through further research in a number of areas. One major drawback of the experimentation done in this paper was the small number of test instances. This was due to a number of factors including limited resources and time. Future work should seek to perform many more tests to obtain more comprehensive and definitive results.

In addition future research efforts should include experimentation with ways for attackers to detect the peers to which the vendor node is connected. This would allow the attacker to regain some of the benefit of delaying the transmission of TR_A and using more nodes to propagate TR_A. Further, tests should be performed where the attacker is allowed to connect to one or more of these peers. Such tests may reveal the benefit of step (5) of the hybrid approach.

A final area of further research should involve experimenting with different types of peers and different levels of connectivity between peers. In our test environment, peers were chosen at random. Various improvements may be gained through more selective peer connections. For example, some peers such as large miners may be very well connected and offer substantial benefits to detection of race attacks, while others may have few connections and offer little.

X. Conclusion

We have discussed the operation of the Bitcoin network including the transmission and propagation of transactions and the inclusion of transactions in a public ledger known as the blockchain. Further, we have identified a potential vulnerability in the Bitcoin network known as double spending. This vulnerability may be exploited in several different ways including Race attacks, Finney attacks, Vector76 attacks, Brute Force attacks, and 51% attacks. Of these attacks, race attacks are particularly relevant for transactions that must occur quickly before they are confirmed in the blockchain. Several solutions have been proposed to defend against race attacks including listening periods, observers, forwarding double spending attempts, and verifiable code execution. However, each of these falls short in key areas.

Because none of the existing solutions to the race attack are satisfactory, we have proposed a new solution, a hybrid approach. This approach combines aspects from the existing solutions with new features. The approach includes six steps: refuse incoming connections from nodes, establish as many connections as possible with well-known nodes, establish one or more observer nodes, enforce a listening period, modify the Bitcoin client such that transactions sending money to the vendor are not propagated, and modify the Bitcoin client to alert the vendor when an attack is detected.

We have established a test environment to test our proposed hybrid approach consisting of an attacker component that generates valid and invalid transactions, a vendor component that attempts to detect race attacks, and an observer component that assists the vendor in detecting race attacks.

Using the data collected through testing, we have made observations about our hybrid approach and provided an evaluation of it. Our observations have shown that each step in the hybrid approach is beneficial in detecting race attacks, and the combination of all steps results in a solution that is superior to existing methods. Our approach offers a low attack success rate of 5.88%, remains relatively inexpensive and simple to implement, does not require changes to the existing Bitcoin protocol, and does not leave the network open to denial of service attacks.

References

[1] Satoshi Nakamoto. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. [Online]. Available: http://bitcoin.org/bitcoin.pdf

[2] Prabhjot Singh, Srishti Arora, B. R. Chandavarkar, and Neha Agrawal. (2013). Performance Comparison of Executing Fast Transactions in Bitcoin Network Using Verifiable Code Execution. Presented at Second International Conference on Advanced Computing, Networking and Security.

[3] Ghassan O. Karame Elli Androulaki, and Srdjan Capkun. (2012). Double-Spending Fast Payments in Bitcoin. Presented at ACM Conference on Computer and Communications Security. [Online]. Available: http://www.syssec.ethz.ch/research/Bitcoin

[4] Matthias Herrmann. (2012, April 24). Implementation, Evaluation and Detection of a Doublespend-Attack on Bitcoin. [Online]. Available: http://e-collection.library.ethz.ch/eserv/eth:5606/eth-5606-01.pdf

[5] Simon Barber, Xavier Boyen, Elaine Shi, and Ersin Uzun. (2012) Bitter to Better - How to Make Bitcoin a Better Currency. Presented at Proceedings of Financial Cryptography and Data Security. [Online]. Available: http://crypto.stanford.edu/~xb/fc12/bitcoin.pdf

[6] Bitcoin Wiki: Double-spending. [Online]. Available: https://en.bitcoin.it/wiki/Double-spending

[7] Meni Rosenfeld. (2012, December 13). Analysis of Hashrate-Based Double-Spending. [Online]. Available: https://bitcoil.co.il/Doublespend.pdf

[8] David Perry. (2012, October 5). Bitcoin Attacks in Plain English. [Online]. Available: http://codinginmysleep.com/bitcoin-attacks-in-plain-english/

[9] Bitcoin Wiki: Myths. [Online]. Available: https://en.bitcoin.it/wiki/Myths#Point_of_sale_with_bitcoins_isn.27t_possible_because_of_the_10_minute_wait_for_confirmation

[10] Emily Flitter and Douwe Miedema. (2014, February 11). Bitcoin Hit by Denial of Service Attacks as Regulators Prepare Clampdown. [Online]. Available: www.reuters.com/article/2014/02/12/us-usa-bitcoin-idUSBREA1A20X20140212

[11] DeathAndTaxes, forum post, May 2012. [Online]. Available: https://bitcointalk.org/index.php?topic=79090.msg882539#msg882539