A Security Analysis of the Bitcoin Mining Ecosystem

Introduction

Bitcoin and other crypto-currencies have seen a massive growth in popularity and volume of transactions. This growth underpins increased financial risk associated with these currencies. Unlike the old currency paradigm, these new crypto-currencies are very dependent on their underlying technical protocols, algorithms, and implementation.

Crypto-currency systems are technology systems; technical, process, or protocol violations present a huge system risk. To mitigate this risk, crypto-currencies require an in-depth security analysis of the various components that interact with their financial networks. When designing systems, side channels are often overlooked. These side channels can challenge the system's security assumptions and threat models. Security analysis can expose these side channel risks.

This paper explores the common weaknesses and threats against the Bitcoin network. We present our research results from when we reviewed widely used Bitcoin mining software security. Finally, we address the impacts of the vulnerabilities discovered during our research.

In this paper we do not explore any attacks that violate the Bitcoin protocol through code errors. We limit the scope of this paper to discussion of the existing known weaknesses to the Bitcoin protocol and code, design vulnerabilities identified in common mining software, and an impact analysis of those weaknesses and vulnerabilities.

Current Threats and Weaknesses

The primary known weaknesses to the Bitcoin network fall into two categories: Denial of Service and Double Spend Attacks:

- Denial of Service (DOS) violations are just that—the user cannot conduct any Bitcoin transactions. An attacker removes the bitcoin holder's buying power and renders their currency unusable. Besides lowering the victim's holding value, these attacks can result in serious system failures and can have major consequences to critical financial systems.
- A Double Spend Attack can also pose a serious threat to the integrity of the network. In a double spend, a valid transaction is broadcast throughout the network, goods and services are exchanged, and at a later point this transaction is floating—it is not secured in the Blockchain. This floating transaction can be deemed invalid simply by creating a new valid transaction that uses the same inputs and adding it to the Blockchain.

  There are primarily two types of Double Spend Attacks.

  - The first type occurs when a transaction is never added to a Block and is later made invalid by a new transaction. To minimize this risk, good Bitcoin transaction practice recommends users wait for a set amount of confirmations before they can complete an exchange.
  - The second type occurs when a transaction T is added to the Blockchain B, however a longer, valid Blockchain B' appears at a later time. This longer valid Blockchain B' is missing transaction T; instead, a new transaction T' using the same inputs is in its place. The vendor who provided the goods or services to be paid for by transaction T will never receive those bitcoins.

  In both cases the goods and services have been exchanged and the record of the original transaction is no longer valid.

The 51% Attack

There has been a great deal of concern about the 51% attack threat. If a deceitful party has greater than 50% of the network's hash rate, it is easier for them to executing a DOS (denial of service) or Double Spend since the party is mining over half of every new Block:

- To execute a DOS, the deceitful party can maliciously ignore certain transactions on Blocks they have mined. This effectively extends confirmation time and prevents business.
- The malicious party can attempt a Double Spend by redirecting their resources towards an alternative Blockchain. To accomplish this they redirect their resources to mining Blocks privately in order to construct a longer secret Blockchain without a targeted transaction. In this scenario the remaining miners in the network will continue mining and public Blocks will be discovered roughly half as often. This raises the confirmation time to 20 minutes per Block until the next Difficulty adjustment. At a selected time after the goods have been exchanged, the malicious party would release their longer Blockchain to the network and remove the transaction from history.

The primary argument against the 51% attack revolves around an attackers "stake" in the network. An attacker who has gained such a large percentage of the network hash rate has invested a significant amount of capital into dedicated hardware. Since a 51% would be obvious to detect and would lower consumer trust in Bitcoin it would be against the stakeholder's best interests to perform such an attack.

Bitcoin Mining Software Review

For this research, we focused on applications SGMiner, CGMiner, and BFGminer. At the time of our research these applications were some of the most popular mining software available. These applications have a large shared codebase; they are all forked from the same code at various points in time. We selected software targets using the following criteria: commonly used in Bitcoin mining, open source, and written in languages requiring manual memory management.

Protocol Analysis

These clients communicate with pools over the loosely specified Stratum protocol. Since Stratum communicates over TCP and contains no form of encryption or integrity, this permits a variety of MITM attack styles such as a strategically located attacker impersonating a pool or client and easily constructing any message.

Stratum is a JSON based protocol; it defines a total of five unique request and response messages. We analyzed the miner source code and discovered that several undocumented messages were recognized as well. These undocumented messages include:

- "client.show_message" which directly records a string into client logs with no sanitization.
- "client.reconnect" which enables a pool to direct a miner to a separate pool with no user intervention required.

These features, coupled with the lack of transport layer security, pose a serious risk to a miner's system and investment. For example, an attacker in the middle of the connection can redirect a miner to their private pool. The attacker can then collect the rewards of the stolen processing power.

Fuzzing

We then used Peach Fuzzer to fuzz the clients and discovered three unique bugs in the clients' Stratum message parsing:

- CVE 2014-4501 describes an attacker's ability to overflow a stack buffer via a long URL argument in the "client.reconnect" message.

- CVE 2014-4502 enables an attacker to send a large or negative nonce length parameter to the client which causes the miner to calculate an insufficient buffer size for new Blocks and overwrite heap memory.

  Both CVE 2014-4501 and CVE 2014-4502 initiate a DOS on the targeted application and in certain cases can lead to remote code execution on the victim machine. These vulnerabilities were discovered in all three of the tested clients and have since been patched (after responsible disclosure) in the following version releases: SGMiner (4.2.2), CGMiner (4.3.5), and BFGMiner (4.1.0).

- CVE 2014-4503 was discovered in SGMiner and early versions of CGMiner (3.7.2). An attacker in the middle of a connection can send a "mining.notify" message with malformed parameters to the client.

  When certain parameters are not valid strings of hex characters, CVE 2014-4503 will cause application errors and early termination, allowing an attacker to initiate an application DOS. This vulnerability has been patched in the current version of SGMiner (4.2.2) as well.

Impact Analysis

The vulnerabilities we discovered pose a serious threat to the Bitcoin network and its users. A miner communicating via Stratum presents a series of possible scenarios in which a user is at risk. These include:

- An attacker can sniff the cleartext credentials in the mining.authorize message. These credentials may be used elsewhere across the internet and may lead to account compromise.
- An attacker in the middle of a connection can replace the Bitcoin address in the username field of a mining.authorize message with their own to steal the users' payouts from the pool.
- An attacker can spoof a "client.reconnect" message from the pool to redirect the miner to a private pool. This reconnection would not be initially obvious to the users and the pool would not need to payout any shares of the Block rewards.
- An attacker can spoof a message from a pool containing a malicious payload related to one of the discovered CVE's to initiate a client DOS. If this is done en mass, this can reduce mining competition from the attacker's pool, or increase their relative share of the network hashrate and make it easier to execute a successful double spend.
- An attacker or malicious pool can send a message containing a malicious payload that remotely executes code on a victim's machine. This can be used to install malware such as rootkits and keyloggers. If the mining system is also used as a desktop, this can lead to various compromises including stolen passwords, credit card numbers, and banking information.

The vulnerabilities mentioned above also increase the likelihood of a 51% attack from a party with little stake in the mining equipment. An attacker who can compromise mining software and redirect hashes to their own pool may not have invested any capitol into using these exploited devices and may have little concern over the attack's effect on the currency. For this reason, trust in stake holders is a dangerous assumption to make when securing the Bitcoin network.

In conclusion, all applications that support the Bitcoin network should be subjected to regular security testing. As crypto-currency protocols evolve and grow, their applications' software lifecycle must include fuzz testing, code reviews, and design reviews. Technologies like Getblocktemplate (a popular alternative to the Stratum protocol) must be reviewed as well as any new crypto-currency features (such as multi-sig transactions, stealth addresses, and coinjoin).

About Authors

Mick Ayzenberg is the primary author of this paper. Mick is a Security Consultant at Deja vu Security where he focuses on application penetration testing and security fuzz testing. His current research involves crypto-currencies and the security of these payment networks. He has an undergraduate degree from the University of Washington.

Adam Cecchetti is a founding partner of Deja vu Security. His research interests focus on machine learning, hardware security, and automated exploitation. Adam has a graduate degree from the Carnegie Mellon University.

Akshay Aggarwal is a founding partner at Deja vu Security. His primary areas of interest are security of cyber and gaming economies, scalable security testing, and securing the Internet of Things. He has a graduate degree from the University of California at Davis.