Eclipse Attacks on Bitcoin's Peer-to-Peer Network

Presentation by Brett Holden

Why I chose this paper

- We learned a lot about blockchain integrity and consensus
  - What about integrity of the P2P network Bitcoin uses to communicate?
  - What if we just want to manipulate one person’s view of the blockchain?

What is an Eclipse Attack?

- Monopolization of all connections to and from a victim bitcoin node
  - How?
    - Adversary controls sufficient number of nodes
- Attack exploits bitcoin’s mining and consensus system
  - Double spending
  - Selfish mining
  - Adversarial forks in the blockchain

My General Approach

- Start with some major concepts and how they are related
  - Primarily
    - Why bitcoin’s P2P network is vulnerable
    - How eclipse attacks work
    - Countermeasures
- Other interesting things
  - Technical Details
- Won’t discuss
  - Quantifying resources required
    - Probabilistic analysis
    - Monte Carlo simulations
    - Measurements and experiments with live bitcoin nodes

How are Eclipse attacks performed?

- Monopolize victim's incoming/outgoing connections → manipulate their view of blockchain
- Traditional claim by Nakamoto is that bitcoin is secure against attackers as long as attackers control less than half of the networks computational power
  - This incorrectly assumes that all nodes have perfect information
    - I.e. they can observe all proof-of-work done by their peers
  - Much research asserting bitcoin's proof-of-work protocol but little attention given to peer-to-peer network used to exchange information between nodes

Overview of Bitcoin P2P Network

- Network is bundled with bitcoind / Satoshi client
  - Designed to be open, decentralized, independent of public-key infrastructure
- No cryptographic authentication between peers
  - Nodes are simply identified by their (public) IP address
- Outgoing Connections: Each node runs a randomized protocol to form long-lived outgoing connections
  - Also propagate and store addresses of other prospective peers
- Incoming Connections: Nodes with public IPs (most independent nodes) accept up to 117 unsolicited incoming connections from any IP address.
- Nodes exchange views of blockchain with their incoming and outgoing peers

Types of Eclipse Attacks

- Specific goals
  - Force victim to waste computational power on obsolete views of blockchain
  - Coopt the victim's computational power for your own use
- Types of Attacks → classification based on attackers resources
  - Off-path (Botnet)
    - Attacker controls **diverse** range of hosts but not key infrastructure between victim and rest of network
  - Infrastructure (ISP, company, nation-state)
    - Attacker controls **contiguous** range of hosts

Eclipse Attack Details

- Attack steps
  - Rapidly/repeatedly form unsolicited incoming connections to the victim from attacking nodes
    - Connections relay bogus network information about other nodes
  - When victim restarts their is a high probability all eight of the outgoing connections will be to attacker nodes
    - Because we gave them fake information about other nodes → connections to other nodes will fail
  - We are now dominating all incoming and outgoing connections on the victim node
- Primary challenge → controlling sufficient number of nodes
  - Why? Because attack relies on low-rate TCP connections
  - Off-Path(Botnet/Diverse addresses) attack requires fewer nodes
  - Infrastructure(Contiguous addresses) requires more nodes
    - Hundreds of entities have sufficient addresses at their disposal

Initial Countermeasures

- Current and Ineffective Countermeasures
  - Disabling incoming connections
  - Choosing outgoing connections to well-known, well-connected peers
- Problems
  - No new incoming connections → new nodes cannot join the network
  - How to choose reputable peers?
    - Make private network?
      - Then how do you ensure decentralized computational power to prevent mining attacks?
- Papers goal for countermeasures
  - Increase difficulty of eclipse attacks while remaining decentralized and transparent
- All the major concepts have now been explained → questions?

Bitcoin P2P Information Storing

- Tried Table
  - Stores peers to whom node has established connection (incoming or outgoing)
  - 64 buckets, 64 addresses per bucket
- New Table
  - Stores peers to whom node has not established connection (incoming or outgoing)
  - 256 buckets, 64 addresses per bucket
- Hashing based on IP address used to map addresses to buckets
- DNS Seeders and ADDR messages are used to send information about peers

Selecting Peers for Outgoing Connections (1)

- New outgoing connection formed when
  - Node restarts
  - Outgoing connection is dropped by the network
  - Blacklisting condition is met (only condition in which a node will deliberately drop a connection)
- First decide whether to pick from tried or new

$$\Pr[\text{Select from } \texttt{tried}] = \frac{\sqrt{\rho}(9-\omega)}{(\omega+1)+\sqrt{\rho}(9-\omega)}$$

- p - ratio between number of addresses in tried and new
- w - is the outgoing connection number (possible values are 0,1,2,3,4,5,6,7)

Selecting Peers for Outgoing Connections (2)

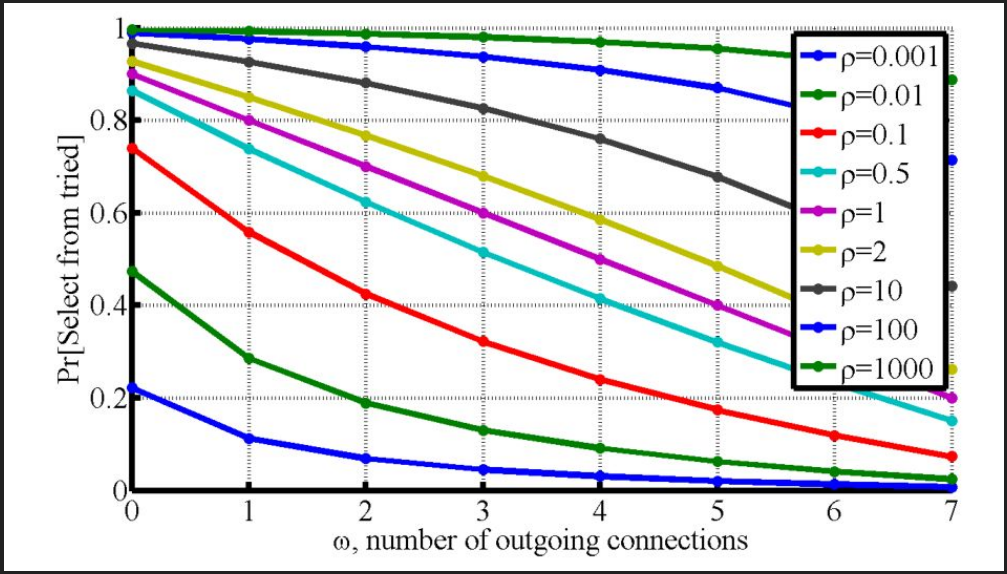
- Next, select a random address from the selected table
  - Bias towards newer timestamps
  - Choose a random non-empty bucket from table
  - Choose a random position in that bucket
    - If there is an address in that position return the address with probability

$$p(r, \tau) = \min(1, \frac{1.2^r}{1+\tau})$$

      - r - number of addresses rejected so far
      - t - difference between the address's timestamp and the current time measured in ten minute increments
    - Else try another random non-empty bucket from the table

Selecting Peers for Outgoing Connections (3)

- Address more likely to be selected from tried when
  - Few outgoing connections
  - Tried table is large



More Eclipse Attack Details

- Attacker populates tried table with addresses for its attack nodes
- Attacker overwrites new with trash/bogus addresses
  - Such as addresses in 252.0.0.0/8 which are available or reserved for future use by IANA
- Attack continues until victim restarts at which point all the incoming/outgoing addresses will be to the attacker’s nodes

Populating Tried and New Tables

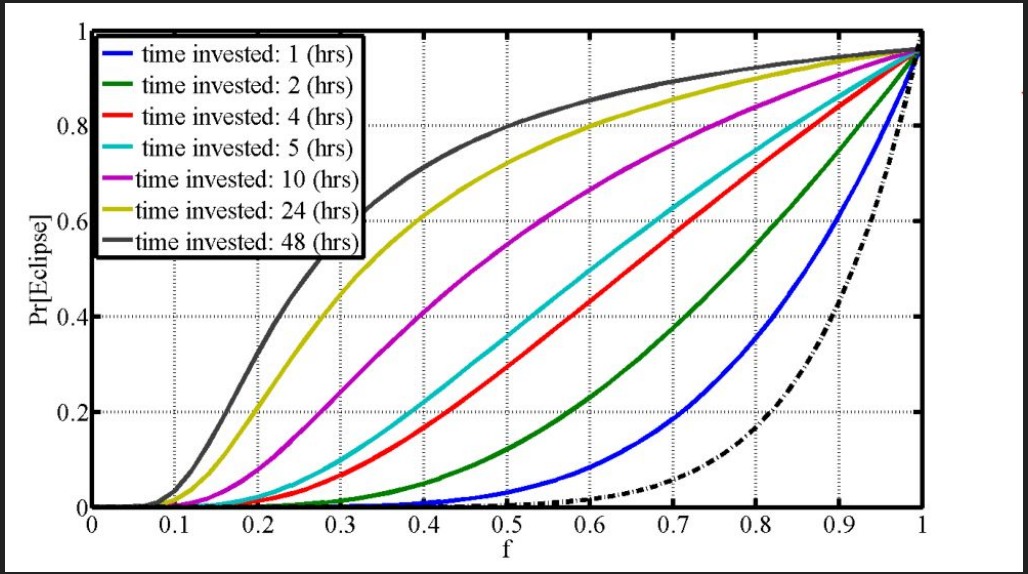
1. Addresses from unsolicited incoming connections are stored in node's tried table
    a. Simply need to connect to victim
    b. Bitcoin eviction principle means that newer attackers addresses will evict older legitimate addresses
2. Nodes accept unsolicited ADDR messages
    a. Nodes insert ADDR messages directly into the new table without testing connectivity
    b. Makes it easy to fill new with trash addresses
        i. Of course better to use trash addresses than legitimate attacker addresses because it saves resources
3. Nodes rarely solicit network information from peers and DNS seeders
    a. This means little is done to counteract the flood of information poisoning tried and new tables

Restarting the Victim

- Attack requires victim to restart so it can connect to attacker addresses
- Restarts are caused by
  - ISP outages
  - Power failures
  - Upgrades
  - OS failures/attacks
  - Software update
  - DDoS
  - Memory exhaustion
  - Packets of death (have been found for bitcoind)
- Average public IP address has 25% chance of going offline after 10 hours

Selecting Outgoing Connections (success rate)

- A lot of math we don’t have time for, the main observation is as follows



- F is the fraction of adversarial addresses

Enhancements

- Avoid OS-mandated TCP connection limits by using a custom TCP stack
  - Zmap - https://github.com/zmap
    - Minimal system resources required, bandwidth is your only bottleneck
- Experiments demonstrate that bitcoin will allow multiple connections from the same IP address
  - This is done to allow multiple nodes behind the same gateway / public address to connect to the same node

Countermeasure - Deterministic random eviction

- Reduces the number of addresses stored in tried
- In addition to hashing each address deterministically into a single bucket, also hash deterministically into a single slot of the bucket
  - Prevents attacker from repeatedly storing the same address in multiple rounds

Countermeasure - Random Selection

- Reduces advantage of newer timestamps and increases time spent on replacing tried and new
- Instead of using prior equation for selecting outgoing connections simply randomly select and address from tried and new.

Countermeasure - Test before evict

- Reduces the rate at which you are able to replace addresses in tried
- Before storing and address in a deterministically-chosen slot in a bucket in tried check if there is an older address stored in the slot.
  - Only replace the older address if a connection to it fails

Countermeasure - Feeler Connections

- Method for cleaning out trash from new and increases likelihood of a fresh address that in tried that will be online when a node restarts
- Perform periodic outgoing connection tests to random addresses in the new table
  - If the connection succeeds move the address into the tried table
  - Otherwise evict the address from new

Countermeasure - Anchor Connections

- Same purpose as feeler connection
- Store an anchor table which records addresses of outgoing connections and the time of first connection to the address
- Establish first two connections to oldest anchor addresses when a node restarts

Countermeasure - More Buckets

- Method for increasing difficulty to poison victim’s tried and new tables
- Simply add more address space to the buckets
  - Assuming presence of legitimate peers it will increase the number of addresses the attacker needs

Countermeasure - More Outgoing Connections

- Increases difficulty of monopolizing victims connections
- Already implemented by some mining pools

Countermeasure - Ban unsolicited ADDR messages

- Prevents attacker from flooding new table with trash addresses
- The ADDR message requested upon establishing an outgoing connection already contains a significant number of addresses

Countermeasure - Diversify incoming connections

- Simply limit the number of connections from a single public IP address
  - This prevents the use of tools like zmap which create an absurd amount of outgoing TCP connections from few addresses

Countermeasure - Anomaly Detection

- Signatures of an Eclipse attack
  - Large number of short-lived incoming TCP connections from diverse IP addresses
  - Connections that are large ADDR messages
  - Messages contain trash IP addresses
- Such detection could force attackers to lower rate of attack or use legitimate addresses to fill the new table

The Result

- Bitcoind updated in early 2015 to include three of the proposed countermeasures
  - Deterministic random eviction
  - Random selection
  - More Buckets
- My two cents
  - Not sure this increased the difficulty of Eclipse attacks enough
  - I expect new flavors of Eclipse attacks in the coming years

Paper Reference

https://eprint.iacr.org/2015/263.pdf

Questions