Recent DDoS and Bitcoin Attacks Exploiting Internet Routing

September 23, 2019

Min Suk Kang
Assistant Professor, Computer Science
National University of Singapore
Knowledge of Internet routing/topology => *stronger* and *stealthier* attacks

   ✓ Knowing how *routing protocol* (i.e., BGP) works *in practice* => *stronger DDoS* attacks

II. Stealthier Bitcoin partitioning attacks [IEEE S&P 2020]
   ✓ Knowing the *Internet topology* => *stealthier Bitcoin* attacks
**Transit-link** DDoS attack: a powerful type of volumetric DDoS attack (distributed denial of service)

Traditional: volumetric attack traffic targeting end servers

Non-traditional: volumetric attack traffic targeting transit links

**Academic studies:**
Coremelt attack (ESORICS ‘09)
Crossfire attack (S&P ‘13)

**Real incidents:**

2013

Spamhaus DDoS grows to Internet-threatening size

More than 300 Gb/s of traffic aimed at the anti-spam site’s hosting

By Alexander J. Martin 9 Nov 2015 at 16:10

2015

ProtonMail DDoS wipeout: Day 6. Yes, we're still under attack

Maybe if you hadn’t paid the ransom to the wrong attackers it would be over
Handling transit-link DDoS attack is **challenging**

**Indistinguishable**
low-rate traffic

Victims are **indirectly** affected
Transit-link DDoS attacks: a long-time open problem

Crossfire attack (Kang et al.)

CoDef defense (Lee et al.)

LinkScope (Xue et al.)

SPIFFY (Kang et al.)

NetHide (Meier et al.)

RADAR (Zheng et al.)

Partial solutions

Routing Around Congestion (RAC) (Smith et al. S&P’18)

Not available in the current Internet

STRIDE (Hsiao et al.)

SIBRA (Basescu et al.)

Adaptive attack against RAC (Tran et al. S&P’19)

“Complete and readily deployable solution!”

Coremelt attack (Studer et al.)

2009

2013

2014

2016

2018

2019
Background: How **BGP routing** works?

**Border Gateway Protocol (BGP)**

- **No control over traffic path by design**
- **Loop-free AS-path**

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>{X, Y, Z, D}</td>
<td>{X, Y, Z, D}</td>
</tr>
<tr>
<td>{X, Y, Z, D}</td>
<td>{X, Y, Z, D}</td>
</tr>
<tr>
<td>{Y, Z, D}</td>
<td>{Y, Z, D}</td>
</tr>
<tr>
<td>{D}</td>
<td>{D}</td>
</tr>
</tbody>
</table>

Traffic path:
- BGP propagation
- Traffic forwarding
Routing Around Congestion (RAC): Rerouting using BGP poisoning [Smith et al., S&P ’18]

Goal: reroute to avoid AS W

AS collaboration is not needed!

Victim destination

BGP poisoning message

Loop detected!

Original path

Detour path

Switch to detour path

Critical source
Will **RAC** defense still work against *adaptive* attackers?
**Adaptive detour-learning** attack [Tran et al. S&P 19]*

**Goals:**
(1) To detect rerouting in real-time
(2) To learn new detour path accurately
(3) To congest new detour path (see the paper)

**Capabilities:**
- Same botnets used in transit-link DDoS attack

Adaptive detour-learning attack: (1) how to *detect* rerouting in *real-time*

![Diagram showing the adaptive detour-learning attack process. The diagram illustrates the original path from AS D to AS Z, the detour path through AS W, and the critical source AS I. The victim destination is AS Y, and the adaptive adversary is represented by a character.]
Adaptive detour-learning attack:
(2) how to learn detour path accurately

Challenge: Which is more accurate route measurement of actual detour path? (3) congest detour path (see the paper)

Solution: Prioritize measurement from bot closer to traffic source

Detour path closer AS
(e.g., shorter AS-path)
Adaptive detour-learning attack:
(2) how to learn detour path accurately

(3) congest detour path

**Challenge:** Which is more accurate route measurement of actual detour path?

**Results:** 94% of learned detour paths are correct

**Solution:** Prioritize measurement from bot closer to traffic source
What if **RAC** defense is *aware of* the adaptive detour-learning attacks?

Can an adaptive RAC defense handle *adaptive* detour-learning attacks?

“**adaptive**” RAC defense?

“**adaptive**” attack
**Necessary condition** for preventing detour-learning attack

Detour path must be **isolated**!

How to isolate?

*Poison all peers* of ASes on detour path!
Detour path isolation => poisoning too many ASes

Thousands ASes should be poisoned
But why?

Tier-1 or large Tier-2 on the detour paths (more in the paper)
Can we *poison* that *many* ASes?

**Specification**
- up to 2034

**Implementation**
- up to 255

**Configuration**
- up to 30-50

CDF

Number of ASes that should be *poisoned*
Confirmed: ISPs do not support poisoning > 255 ASes

Number of observed BGP messages slowly decrease in frequency 50x drop in frequency

Number of ASes seen in a BGP message
**Confirmed**: ISPs do **not** support poisoning > 255 ASes

Poisoning > 1,000 ASes is *nearly impossible*

=> Detour path isolation is *infeasible*

=> *Detour-learning attack is almost always possible*
Transit-link DDoS attacks still remain an open problem.

**Crossfire attack** (Kang et al.)
- **CoDef defense** (Lee et al.)
- **Routing Around Congestion (RAC)** (Smith et al. S&P’18)
  - “Complete and readily deployable solution!”
- **Not available in the current Internet**
  - STRIDE (Hsiao et al.)
  - SIBRA (Basescu et al.)
- **Partial solutions**
  - LinkScope (Xue et al.)
  - SPIFFY (Kang et al.)
  - NetHide (Meier et al.)
- **RADAR** (Zheng et al.)
- **Adaptive attack against RAC** (Tran et al. S&P’19)
- **Not available in the current Internet**
- **NetHide** (Meier et al.)
- **STRIDE** (Hsiao et al.)
- **SIBRA** (Basescu et al.)
- **Routing Around Congestion (RAC)** (Smith et al. S&P’18)
  - “Complete and readily deployable solution!”
Two Lessons Learned
Lesson 1

Hacking the current Internet routing for routing around congestion is a \textit{flawed} idea!
✓ Adaptive attacks are possible and much stronger than adaptive defense

✓ Mitigation is hard
Lesson 2

Analysis of protocol *specifications alone* is *insufficient*!
Knowledge of Internet routing/topology => stronger and stealthier attacks

   ✓ Knowing how *routing protocol* (i.e., BGP) works *in practice* => *stronger DDoS* attacks

II. Stealthier Bitcoin partitioning attacks [IEEE S&P 2020]
   ✓ Knowing the *Internet topology* => *stealthier Bitcoin* attacks
Bitcoin relies on the underlying networks

Consensus protocol
Peer-to-peer network
Inter-domain network
**Partitioning attack** in Bitcoin

- isolate one or more Bitcoin nodes from the rest of the network
- lead to consensus attacks
  - e.g., double-spending
- can even bring down entire Bitcoin!
Previous attack: *routing manipulation* to partition Bitcoin’s peer-to-peer network

- Proposed by Apostolaki et al. [IEEE S&P 2017]

Legend:
- Original peer connection
- Hijacked peer connection
- Benign Bitcoin node

(BGP-hijacking)
Lie: “I have a better path to V”

=> **hijack** p2p conn. of victim
=> can **split** Bitcoin when hijacking ~100 nodes

Previous attack: *routing manipulation* to partition Bitcoin’s peer-to-peer network

- Proposed by Apostolaki et al. [IEEE S&P 2017]

---

**Legend:**

- Original peer connection
- Hijacked peer connection
- Victim node \(V\)
- Benign Bitcoin node \(M\)
- Attacker ISP \(H\)

(BGP-hijacking)

Lie: “I have a better path to \(V\)”

=> hijack p2p conn. of victim

=> can **split** Bitcoin when hijacking ~100 nodes

---

ASes (e.g., large ISPs) **can** do it.

✓ Question: “**Do they really launch this Bitcoin partitioning attacks in practice?**”
Bitcoin hijacking observed in practice (Sept – Dec 2018)

• Result of our analysis of 48 billion routing data

All 7 cases are due to **misconfigurations**

<table>
<thead>
<tr>
<th>Case</th>
<th>Date of incident</th>
<th>No. of Bitcoin hijacking (#Type-1, #Type-2)</th>
<th>AS Number of invalid BGP message generator</th>
<th>No. of victim AS</th>
<th>Propagation hops of BGP messages (avg±std dev)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Oct. 17, 2018</td>
<td>57 (0, 57)</td>
<td>14259</td>
<td>3</td>
<td>1.0±0.0</td>
</tr>
<tr>
<td>2</td>
<td>Nov 26, 2018</td>
<td>35 (36, 0)</td>
<td>17639</td>
<td>20</td>
<td>1.47±0.74</td>
</tr>
<tr>
<td>3</td>
<td>Nov 27, 2018</td>
<td>208 (0, 208)</td>
<td>8928</td>
<td>66</td>
<td>1.0±0.0</td>
</tr>
<tr>
<td>4</td>
<td>Nov 27, 2018</td>
<td>214 (0, 214)</td>
<td>8928</td>
<td>65</td>
<td>1.0±0.0</td>
</tr>
<tr>
<td>5</td>
<td>Nov 27, 2018</td>
<td>217 (0, 217)</td>
<td>8928</td>
<td>67</td>
<td>1.0±0.0</td>
</tr>
<tr>
<td>6</td>
<td>Dec 10, 2018</td>
<td>54 (0, 54)</td>
<td>14259</td>
<td>3</td>
<td>1.0±0.0</td>
</tr>
<tr>
<td>7</td>
<td>Dec 18, 2018</td>
<td>47 (0, 47)</td>
<td>14259</td>
<td>2</td>
<td>1.0±0.0</td>
</tr>
</tbody>
</table>

**No** attacks in practice. Why?

• Route manipulation is **immediately visible** to the public
• Attacker’s **identity** (AS number) is **revealed**
Bitcoin hijacking observed in practice (Sept – Dec 2018)

• Result of our analysis of 48 billion routing data

No attacks in practice. Why?
• Route manipulation is immediately visible to the public
• Attacker’s identity (AS number) is revealed
The *Erebus* attack: a *stealthier* Bitcoin partitioning attack (Tran et al., IEEE S&P 2020)

**Challenge 1:**
Is there enough Bitcoin nodes that the attacker can use?

**Challenge 2:**
How to change the target node’s existing peers?

- **Idea:** Force the victim node connects to peer \{C,D\} only:
  - Attacker AS is the man-in-the-middle of all peer connections!
  - No route manipulation => *nearly invisible*!
Challenge 1: finding Bitcoin nodes to force the victim node to connect to

- The attacker AS has find at least 125 Bitcoin nodes whose communication routes with the victim include the attacker
Erebus: using “shadow” Bitcoin IPs

**Shadow IPs** are:
- valid IP addresses whose victim-to-IP address routes include the attacker AS
- **not** necessarily used by real Bitcoin nodes!
Erebus attacker can find millions of (virtual) shadow Bitcoin nodes

• If attacker AS is big enough (e.g., rank 100), **millions** of virtual shadow IPs are available

• Shadow IPs are **geographically well distributed**
  • e.g., attacker is Singtel, victim Bitcoin nodes in AWS
  • Look normal even to cautious Bitcoin nodes
Erebus attacker can find millions of (virtual) shadow Bitcoin nodes

• If attacker AS is big enough (e.g., rank 100), millions of virtual shadow IPs are available

• Shadow IPs are geographically well distributed
  • e.g., attacker is Singtel, victim Bitcoin nodes in AWS
  • Look normal even to cautious Bitcoin nodes
Challenge 2: Creating victim-shadow peering connections

"IPs I’ve learned from peers"

"IPs I’ve actually connected to"

Tables for IP addresses

new

tried

"Choose 1 IP randomly from the two tables for outgoing connection"

victim node

Since Bitcoin Core v0.10.1 (2015):

✓ Flooding new table is very hard
✓ Flooding tried table is impossible

~ 3K botnets

Eclipse attack
(Heilman et al., USENIX Sec’15)
Erebus: how to fill the two tables?

Fill with **millions of shadow IPs and wait patiently**!

- Can fill the **new** table easily with widely **diverse** IP addresses
- Can fill the **tried** table by patiently waiting for trickle-downs (e.g., one IP moved from **new** to **tried** every two minutes)
Erebus: trickle-down attack strategy

**new table**

- All IPs
- Reachable IPs
- Shadow (legitimate)
- Shadow (virtual)

**tried table**

- All IPs
- Reachable IPs
- Shadow (legitimate)
- Shadow (virtual)
Patiently attacking a target Bitcoin node
Who can launch the Erebus attack?

• **Global-scale** ISPs
  • AT&T, CenturyLink, NTT, PCCW, DT, ...
  • Can target any Bitcoin node!

• **Large local** ISPs
  • Singtel, China Telecom, ...
  • Can target the majority of Bitcoin nodes!

• **State-sponsored** adversaries
  • Some countries are believed to have direct control over their ISPs

(https://geographicalimaginations.com/2015/09/29/visualising-the-invisible/)
What about *other cryptocurrencies*?

- Bitcoin peer-to-peer networking stack is widely replicated
  - **34 out of top-100** cryptocurrencies are potentially vulnerable to Erebus
Countermeasures?

• The Erebus attack exploits the **topological advantage** of being large ISPs, *not* any specific bugs ⇒ countermeasures are *not* straightforward

[C1] **Table size reduction**  *Awaiting*
Reducing the size of the two tables storing peer IPs makes Erebus attack less effective because the adversary has much larger bandwidth capability and significantly more IP addresses than legitimate peers.

[C2] **More outgoing connections**  *Awaiting*
Increasing the number of outgoing connections (e.g., from 8 to 16) also makes Erebus attack significantly harder to occupy all the outgoing connections.

[C3] **Selecting peers with AS topology information**  *Being implemented by Bitcoin core team*  *Pull request #16702*
Incorporating AS topology in the peer selection can make attack becomes harder or impossible for the adversaries with IPs distributed in a large number of prefix groups but hosted in a few ASes only.

[C4] **Smarter eviction policy**  *In talks with Bitcoin core team*

Please check:  [https://erebus-attack.comp.nus.edu.sg/](https://erebus-attack.comp.nus.edu.sg/)
Two Lessons Learned
Lesson 1

Large ISPs *can* partition any Bitcoin nodes with *no* routing manipulation (virtually undetectable)
✓ Low rate sending only about 2 IPs/sec

✓ Patiently waiting for a few weeks

✓ Large ISPs can launch this attack today against Bitcoin Core v0.18.0
Lesson 2

Mitigation is hard
✓ Exploiting topological advantage not bugs

✓ Multi-layer problem: routing layer and overlay peer-to-peer network

✓ Some changes may cause other problems

✓ More updates on countermeasures: https://erebus-attack.comp.nus.edu.sg/
Takeaways

   ✓ Knowing how *routing protocol* (i.e., BGP) works in practice => *stronger DDoS* attacks

II. Stealthier Bitcoin partitioning attacks [IEEE S&P 2020]
    ✓ Knowing the *Internet topology* => *stealthier Bitcoin* attacks
       => [https://erebus-attack.comp.nus.edu.sg/](https://erebus-attack.comp.nus.edu.sg/)
Question?

Min Suk Kang (kangms@comp.nus.edu.sg)