iSPY: Detecting IP Prefix Hijacking on My Own

Zheng Zhang, Ying Zhang, Y. Charlie Hu, Z. Morley Mao,
Randy Bush (SIGCOMM 2008)

April 23, 2010
Presented by Jaeyoun Kim

Table of Content

2

1. Introduction – *What's Prefix Hijacking?*

- A special form of DoS Attack - *corrupting Internet routing tables*
  - Bad BGP announcement - Forwarding tables get polluted with bogus route
  - *Malicious AS can send & receive traffic using addresses it does not own*
  - Used for carrying out malicious activities
- Serious threat and *hard to eliminate*
  - Lack of authoritative info. on prefix ownership

3

Critical Requirements

- Prefix Hijack Detection System should satisfy all requirements
  1. Real-time
  2. Accurate
  3. Light-weight
  4. Easy to deploy
  5. Incentive to deploy
  6. Robust in Victim Notification

4

Existing Detection mechanisms

- Infrastructure-based detection
  1. *Control-plane-based only*
     - Easily deployable, yet needs live BGP feeds & fairly inaccurate
  2. *Control plane + Data plane* (joint analysis)
     - Real-time, yet needs live BGP feeds and has vantage point limitation
  3. *Data plane only*
     - Easily deployable, yet has vantage point limitation

  – None of them satisfy all critical requirements

5

2. BGP Prefix Hijacking (3 main types)

1. Regular prefix hijacking
   – Attacker originates route to an existing IP prefix of the victim network *(Partial pollution)*
2. Subprefix hijacking
   – Steals subnet of existing prefix by announcing route for it *(Most networks get polluted)*
3. Interception based hijacking

iSPY addresses the regular prefix hijacking

6

3. Key Observations

- Significant percentage of ASes get polluted
- Probes from prefix-owner are unreachable to many ASes
- *Unique Unreachability Signature of Hijacking*
  - *Can distinguish it from other disruptive routing events* such as link failure and congestion

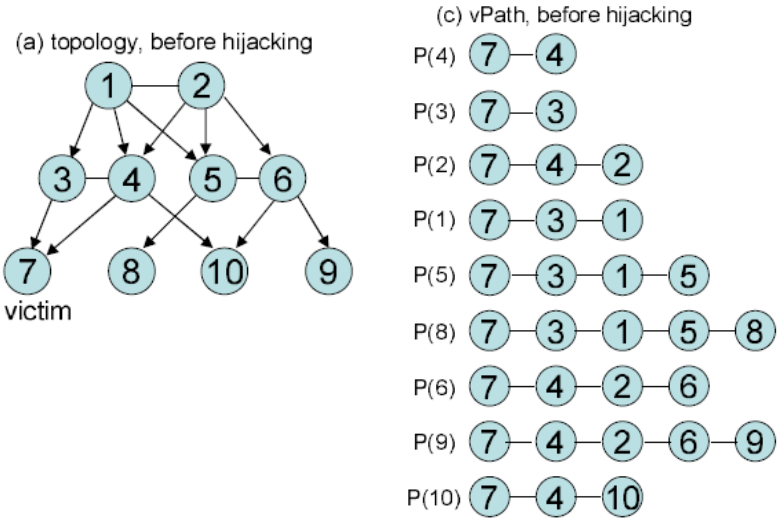
7

3.1 Prefix Owner's View of Reachability

- Capture it as a set of paths called *vPath* (victim's path)
  - Set of AS-level *forward paths* from prefix owner to a specific AS on the Internet
  - traceroute replies will not reach the victim network *(indirectly capture reachability)*
- Networks with multiple prefixes
  - vPath to these prefixes may differ
  - Select any prefix and regard the path to it as path to destination AS

8

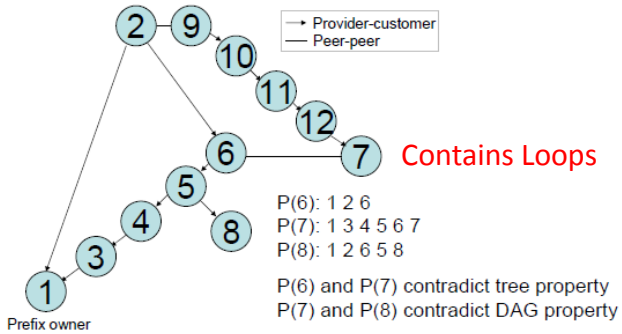
AS Topology Example



9

AS Topology Example (cont'd)

- Cannot compress vPath to *trees* or *Directed acyclic graphs*
  - Due to policy-based routing in the Internet



10

Prefix Owner's View of Reachability (cont'd)

- Case of Potential Route Asymmetry
  - *Destination AS can be reachable* even if *certain ASes* along the forward path to it *are polluted*
  - [a,b,**c**,**d**,e] : polluted c & d returns "*", finally reach e
  - [a, b, **#**, e] : AS-level path may contain # (*uncertain part of AS path*)
- Monitoring Reachability to Transit ASes Only
  - Cost of obtaining paths to many ASes is high
  - All attacks are still covered because *hijack from a stub AS has to pollute its provider transit AS(es)*

11

3.2 Hijack Detection Problem

- Potential Hijack Detection
  - Take periodic snapshots of vPath
  - Compare new snapshot($\mathbf{T_{new}}$) with old snapshot($\mathbf{T_{old}}$) to check for unreachability
  - Possible hijacking when $\mathbf{T_{old}}$ has full reachability and $\mathbf{T_{new}}$ has partial reachability
- Problem
  - Partial reachability ($\mathbf{T_{new}}$) could be *due to other routing anomalies*
  - Need to *analyze the unique characteristics of the gap* between $\mathbf{T_{new}}$ and $\mathbf{T_{old}}$

12

Four Cases - Definition of cuts

Old Path P(d) = {s, u_1, u_2, ... , u_n, d}
New Path P'(d) = {s, v_1, v_2, ... , v_n, d}

1. P(d) remains complete - no cut
2. P(d) becomes partial in T_{new}
   - ui: Last AS in P(d) for which traceroute obatined a reply
   - (u_i, u_i+1) is a cut
3. P(d) changes to P'(d) in T_{new}, and P'(d) is complete - no cut

13

Four Cases - Definition of cuts (Cont'd)

4. P(d) changes to P'(d) in Tnew, and P'(d) is partial
   - v_i: Last AS in P'(d) for which traceroute obtained a reply
   - (v_i, v_i+1) is a cut – if v_i appears in P(d)
   - (v_i, *) is a cut – if v_i does not appear in P(d)

14

Four Cases - Definition of cuts (Cont'd)

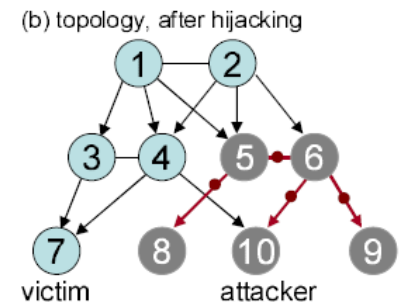
- Denote set of distinct cuts as Ω
- Definition of cuts can handle the cases of uncertain subpaths "#"

Table 2: Examples of cuts under the cut definition.

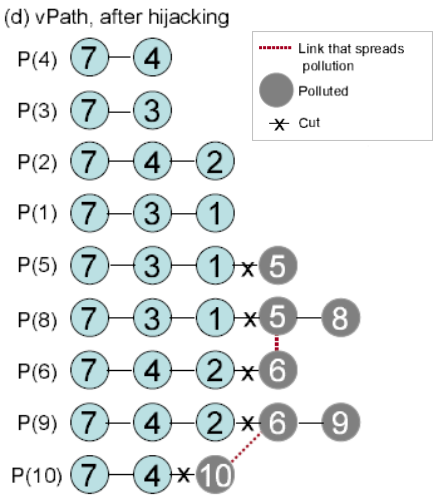
Cut		Current path $P'(d)$		
		abcd	ab#d	ab#
Previous path $P(d)$	abcd	no cut	no cut	bc
	ab#d	no cut	no cut	b#

15

Cuts in vPath – Example



Ω={(1,5), (2,6), (4,10)}
Create 3 cuts



16

3.3 Unreachability Signature of Hijacking

- Size of Ω
  - *Almost Always Large* during ongoing prefix hijack, typically small otherwise
- Rationale
  - Internet topology is *not a tree*
  - *Many peering & multi-homed links*
    - Pollution spreads far and victim network sees many cuts
  - Conventional disruptive Routing Events *(Link failure, congestion)*
    - small cuts, mostly near victim AS

17

3.4 Simulation Validation

- Methodology
  - *Simulate 2,450 hijacking instances*
  - Algorithm on data from RouteViews from *100 vantage points*
  - AS relationship obtained by running *Gao's algorithm*
  - ASes classified into 5: *tier-1, tier-2 transit, tier-2 stub, tier-3+ transit, tier-3+ stub, based on type and number of providers*

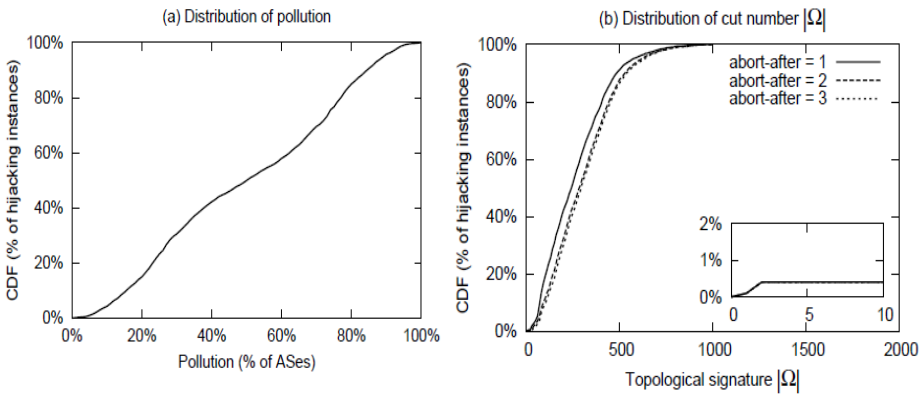
18

Simulation Validation (Cont'd)

- Procedure
  - Compute the forward path P(d) – old vPath
  - Simulate false origin prefix hijacking
  - Compute the forward path P'(d) - new vPath
    - Simulate uncertain non-trailing subpath (#) by aborting traceroute after a *fixed number(abort-after)* of consecutive unreachable hops
  - Calculate the cuts Ω using P(d) and P'(d)
- Limitation
  - Detection delay (due to snapshot durations / start time of hijack and probing rounds)

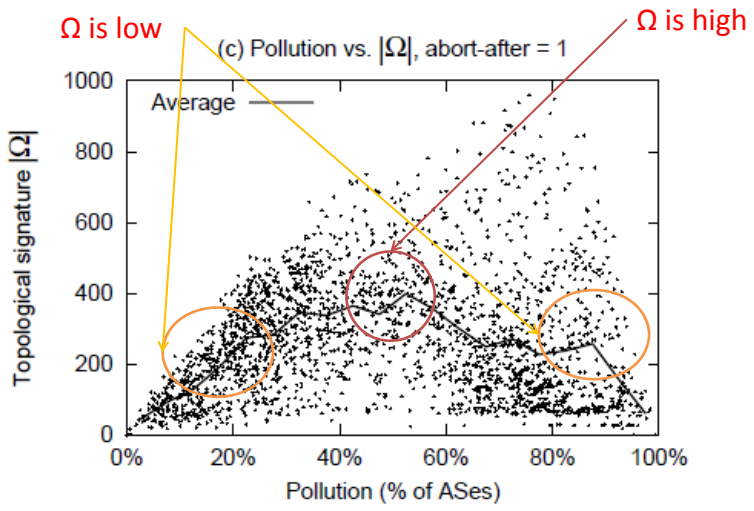
19

Simulation Results



20

Simulation Results (Cont'd)



21

Simulation Results (Cont'd)

- When Ω is small, Ω varies little under different traceroute configuration (abort-after)

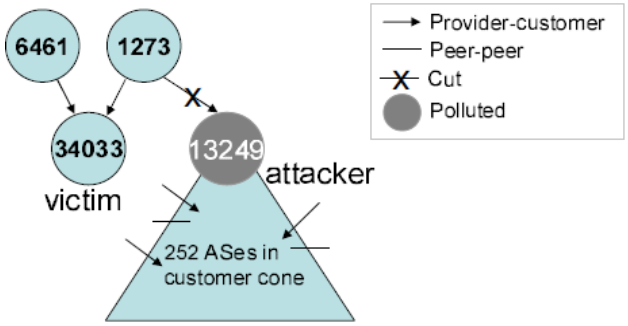
Table 3: The percentage of small $|\Omega|$ instances.

Victim category	Total instances	Small $\|\Omega\|$ instances		
		$\|\Omega\| \leq 5$	$\|\Omega\| \leq 10$	$\|\Omega\| \leq 20$
Tier-1	490	3 (0.61%)	3 (0.61%)	4 (0.82%)
Tier-2 transit	490	1 (0.20%)	1 (0.20%)	1 (0.20%)
Tier-2 stub	490	4 (0.82%)	4 (0.82%)	5 (1.02%)
Tier-3+ transit	490	3 (0.61%)	3 (0.61%)	4 (0.82%)
Tier-3+ stub	490	0 (0.00%)	0 (0.00%)	0 (0.00%)
Any	2450	11 (0.45%)	11 (0.45%)	14 (0.57%)

22

An example hijacking instance with small |Ω|

- AS 13249 hijacks AS 34033
  - Pollution is restricted to the attackers' customer cone
  - Only 144 ASes are polluted
- The inverse case also has a small # of cuts



23

Analyzing Hijacking Instances with Small cuts

- Two Key Rare Conditions for a small cuts
  1. None of attacker's provider(s) is polluted
     - Not likely for randomly picked victim & attacker pairs
     - To satisfy, the victim must also be same provider's customer
  2. Attacker's customers rely heavily on attacker's transit service
     - Not likely unless the customer cone is small

24

3.5 Detecting Known Hijacking Events

Table 4: Cuts in historical hijacking events.

Victim prefix	Victim prefix owner		Attacker	Pollu. (%)	$\|\Omega\|$
64.233.161.0/24	Google	15169	Cogent	31.6	492
63.165.71.0/24	Folksamerica	26913	ConEd.	65.7	458
64.132.55.0/24	OverseasMedia	33477	ConEd.	33.1	176
65.115.240.0/24	ViewTrade	23004	ConEd.	49.4	369
65.209.93.0/24	LavaTrading	35967	ConEd.	16.4	221
66.194.137.0/24	MacKayShields	31860	ConEd.	32.3	261
66.207.32.0/20	ADI	23011	ConEd.	83.0	594
69.64.209.0/24	TheStreet.Com	14732	ConEd.	78.0	658
160.79.45.0/24	RhodesASN	33313	ConEd.	27.5	380
192.251.16.0/24	T&TForex	20179	ConEd.	14.7	170
198.15.10.0/24	TigerFund	5703	ConEd.	86.0	707
204.13.72.0/24	FTENNY	33584	ConEd.	34.6	205
216.223.46.0/24	SDSNY	12265	ConEd.	77.6	606

- All highjacks: $\Omega > 170$
  - though the pollution varies from 14.7% to 86%

25

4. Design – Probing Module Components

1. Probing only transit Ases
   - Reduce the probing cost (23,191 ASes → 3,742 ASes)
2. Live IPs
   - Collect probing candidate IPs from several sources
3. Resolving IP-level paths to AS-level paths
   - Generate *IP-to-AS mapping* using BGP routing tables
   - *Collapse consecutive hops mapped to the same AS*
   - Unresolved hops collapsed to symbol '#'
4. Increasing the efficiency & Robustness of Traceroute
   - Modify *Paris-traceroute* to perform IP-to-AS translation on the fly

26

Evaluation – No Prefix Hijacking

Table 5: Efficiency of iSPY's probing module.

		Five sample sources (by location)					Overall (108 sources)		
		UK	Pitts,US	LA,US	Norway	Japan	min	max	median
①	Avg hops per traceroute	16.6	13.5	17.2	16.7	16.1	10.7	19.9	15.5
②	Probing traffic per round (MB)	1.7	1.4	1.8	1.7	1.7	1.1	2.1	1.6
③	Time per traceroute (sec)	11.3	10.9	11.7	11.0	11.4	9.6	19.5	11.4
④	Probing time per round (min)	17	17	19	17	17	15	29	18
⑤	Bandwidth (KB/s)	1.7	1.4	1.6	1.7	1.6	0.8	2.2	1.5

- 108 PlanetLab nodes (each node probes the 3470 transit Ases)

① determine the efficiency of the whole probing round

② traceroute + ICMP ping + TCP ping

③ 9.6s ~ 19.5s

④ Short turn around time: iSPY can obtain the up-to-date vPath

⑤ Low bandwidth - Light-weight

27

Evaluation - Coverage

	Transit ASes	
	Number	Percent
Traceroute stat		
Probed	3470	100.0%
Reached	3170	91.4%
AS-path completely resolved	2663	76.7%
AS-path incompletely resolved	807	23.3%
Has at least 1 unmapped IP hop	155	4.5%
Has at least 1 unmapped * hop	680	19.6%
Complementary ping stat		
Probed	300	8.6%
Reached	261	7.5%
Complementary TCP stat		
Probed	39	1.1%
Reached	37	1.1%
Traceroute + ping + TCP stat		
Reached	3468	99.9%
AS-path completely resolved	2663	76.7%

- 99% ASes are reached
- 76.7% AS-path completely resolved
  - Due to unmapped hops (*s)

28

Evaluation – Coverage (Cont'd)

Table 7: Coverage of probing on 108 PlanetLab nodes.

	Five sample sources (by location)					Overall (108 sources)		
	UK	Pitts,US	LA,US	Norway	Japan	min	max	median
ASes reached by probing (%)	99.9	100.0	99.9	100.0	100.0	95.6	100.0	99.9
ASes having complete path (%)	79.7	74.5	80.7	82.4	82.4	69.7	85.9	81.0

29

5. Prefix-Owner-Centric Hijack Detection

- Handling Uncertain Subpaths
  - *Calculate the lower bound* of |Ω| (*all uncertain cuts sharing the same starting node are the same)*
  - *Calculate the upper bound* of |Ω| (*each uncertain cut is a different cut)*
  - *Use the lower & upper bound to aid decision making*
- Continuous Decision Making
  - *Continuously stream new vPath data into the decision making module*
  - Can detect hijacking well before all cuts in a complete round of probing are witnessed.

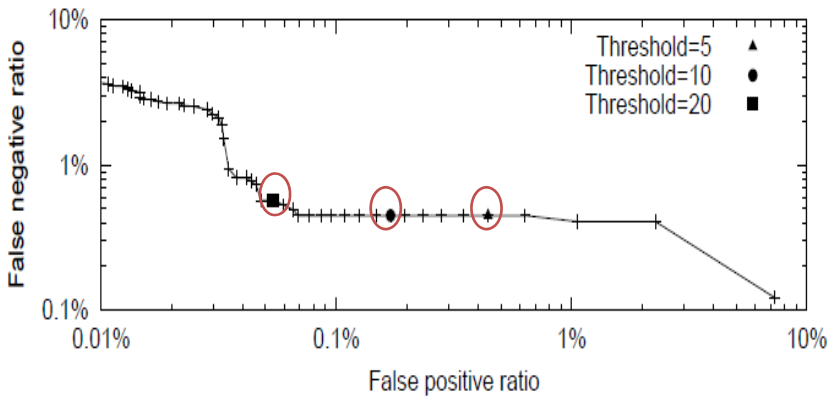
30

6.1 PlanetLab Experiment

- Evaluate the detection false positive ratio
- Detection Accuracy
  - 0.17% alarm (all false positive)
  - Ω did not last for more than one round & No Multiple Origin AS announcement
- Choice of Detection Threshold
  - 10 cuts

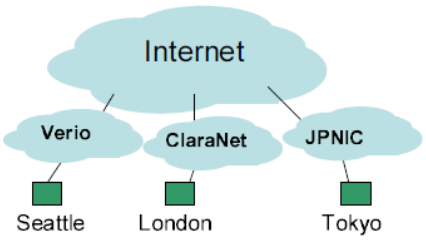
31

PlanetLab Experiment (Cont'd)



32

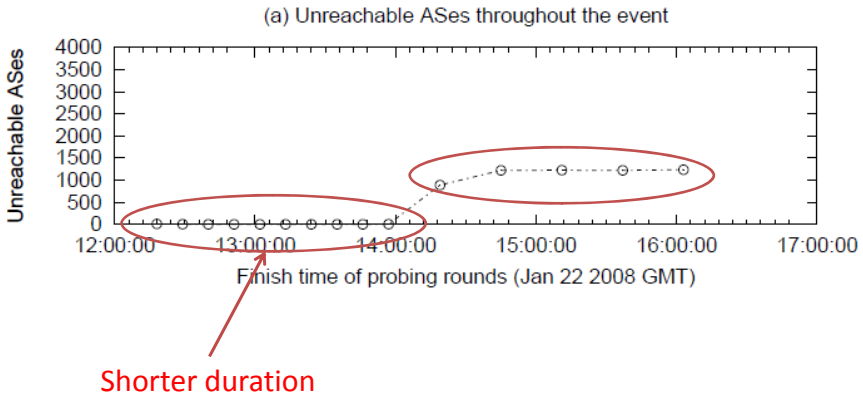
6.2 Hijacking Experimen



- Experiment Setup
  - 3 hosts
  - Launched 15 attacks on their own prefix
  - Allow to inject an anycast prefix from 3 hosts
- Experiment Step
  1. The victim injects the target prefix
  2. Two hours later, the attacker also injects
  3. The attacker withdraws prefix after 2 more hours

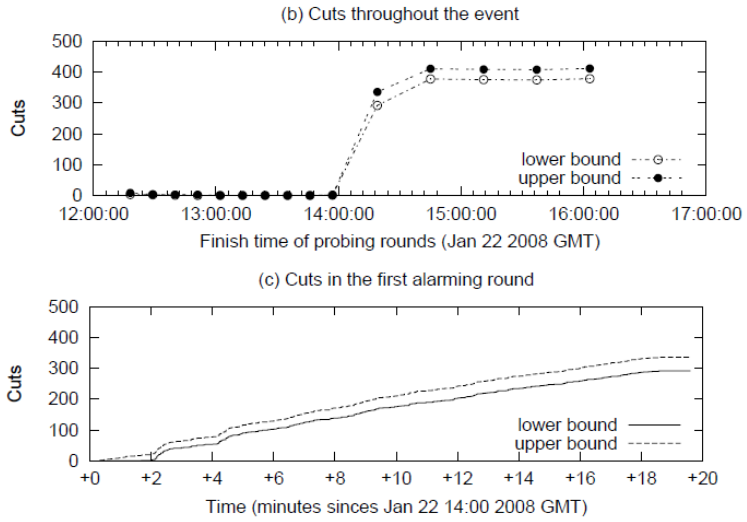
33

of unreachable ASes & cuts



34

of unreachable ASes & cuts (Cont'd)



35

Statics of the 15 hijacking events / Detection Performance

Event #	Scenario					iSPY Performance			
	Victim	Attacker	Hijack start time (GMT)	Pollution (%)	Cuts ([LB, UB])	Detected?	1st alarming round start time (min)	Detection latency (min)	Pollu. at alarm (%)
1	Seattle	London	Jan 22 14:00	35.6	[376, 409]	yes	-2.1	2.1	0.4
2	"	"	Jan 23 20:00	36.0	[383, 415]	yes	-4.0	2.1	0.4
3	"	"	Jan 25 02:00	36.0	[384, 417]	yes	-7.0	2.1	0.3
4	"	"	Jan 26 08:00	36.4	[382, 417]	yes	-4.7	2.7	0.4
5	"	"	Jan 27 14:00	36.1	[376, 409]	yes	-8.3	2.3	0.4
6	"	"	Jan 28 20:00	36.3	[379, 413]	yes	-2.6	2.7	0.5
7	Seattle	Tokyo	Jan 22 20:00	31.4	[205, 231]	yes	-2.7	2.7	0.5
8	"	"	Jan 24 02:00	31.0	[201, 226]	yes	-4.5	2.1	0.3
9	"	"	June 04 02:00	34.4	[219, 246]	yes	0.2	3.1	1.0
10	London	Seattle	Jan 27 02:00	51.3	[331, 376]	yes	-2.9	1.4	0.3
11	"	"	Jan 28 02:00	51.1	[328, 372]	yes	-2.7	1.4	0.4
12	Tokyo	Seattle	June 02 02:00	51.0	[788, 839]	yes	-0.4	3.1	0.4
13	"	"	June 02 06:00	52.3	[805, 855]	yes	-10.9	1.8	0.4
14	"	"	June 03 08:00	51.4	[785, 833]	yes	-5.5	2.3	0.4
15	"	"	June 03 14:00	51.2	[793, 841]	yes	-6.2	2.4	0.3

Large number of cuts

36

Discussion

- Counter Measures by attackers against iSPY
  - Probe Modification
    - Need to manipulate replies to all traceroute probes
  - Pollution shaping
    - Difficult to shape a small-cut pollution
    - Hard to calculate ASes to add to the initial bogus route
- Future work
  - Detection accuracy improvement selecting personalized thresholds by each network
  - Identifying the attacker in real-time

37

Conclusion

- Highly effective prefix-owner-based IP prefix hijacking detection system (iSPY)
  1. Highly accurate
  2. Almost real time detection (1.4 ~ 3.1 minutes)
  3. Lightweight
  4. Easy to deploy for prefix owner
  5. Strong Incentive to deploy
  6. Robust in victim notification (Hijack detection decision made locally)

38