BLC: Blocklist Co-occurrence Analysis for Large-scale IP Network Traffic Flows

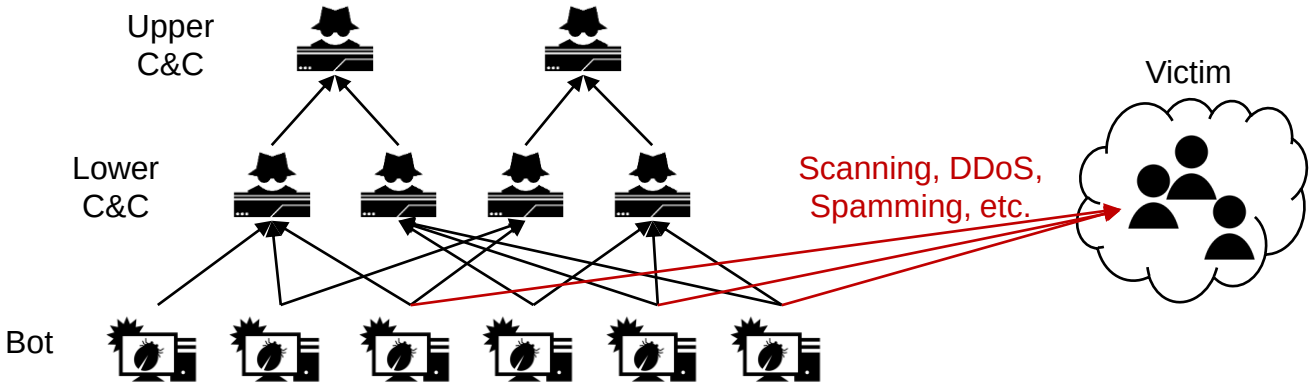
Shosuke Oba[1], Kazunori Kamiya[1], Kenji Takahashi[1], Yasuyuki Hamada[1],
Kazumichi Sato[2], Toshiaki Sudo[3]

[1] NTT Security Holdings,
[2] NTT Network Service System Lab., [3] NTT Communications

Background

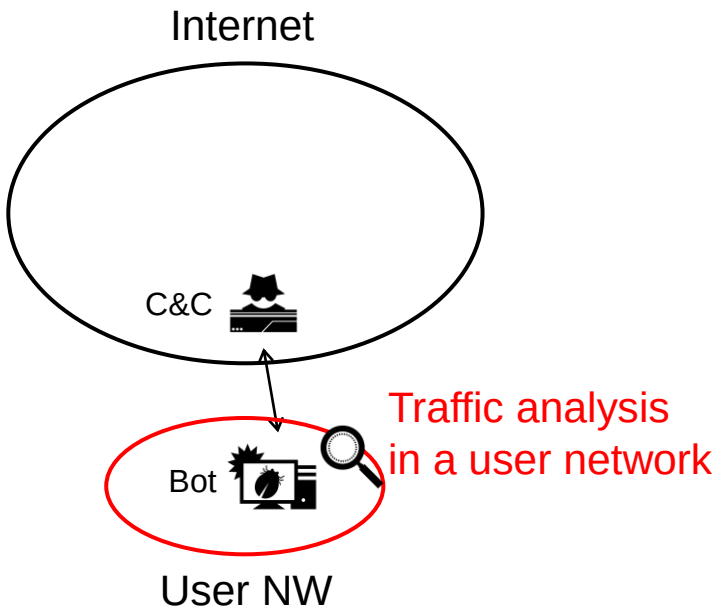
- A botnet is a group of malware-infected hosts that launch various cyberattacks.
- To eliminate botnet threats, as the first step we need to understand the entire picture of botnet infrastructures by
  - detecting components in a botnet, and
  - identifying the relationships between them

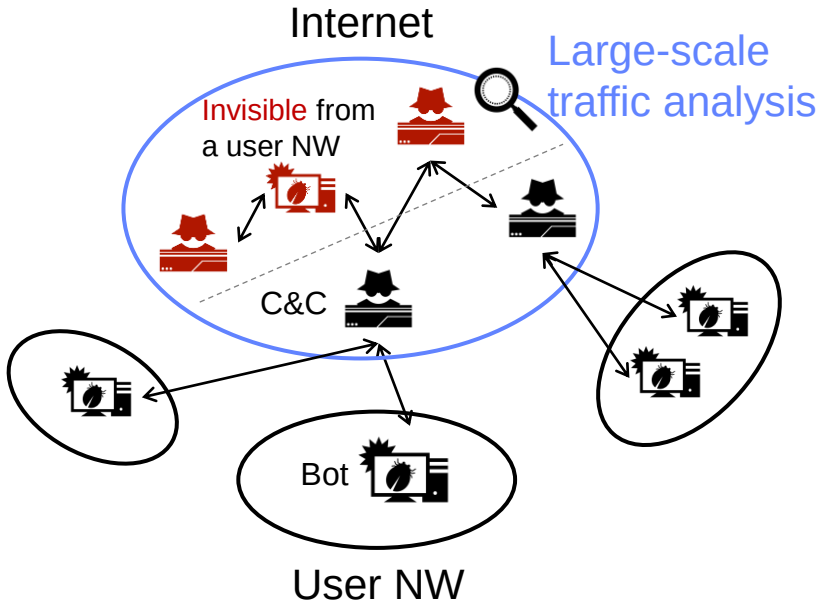
Example of hierarchical botnet structure



Problem Statement 1/3

- Many existing methods focus on traffic analysis in a user network.
- However, they lack the visibility of layered and distributed botnet infrastructure.
- Large-scale traffic analysis at the Internet backbone is necessary.





Problem Statement 2/3

NTT Security Holdings

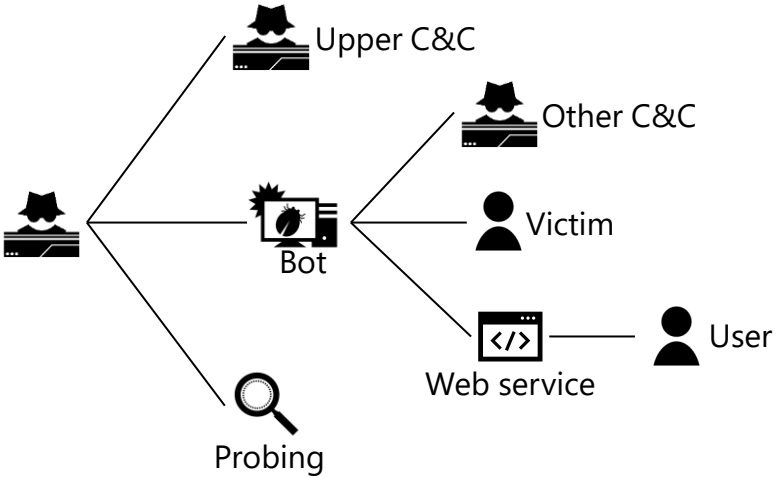
C&C detection by ML

Detect C&C by analyzing their communication behaviors individually but not collectively

- Quantity
- Frequency
- Number of contacts
- Time of day

Graph approach

Grasp a group (or botnet) of collaborating hosts by analyzing communication between them

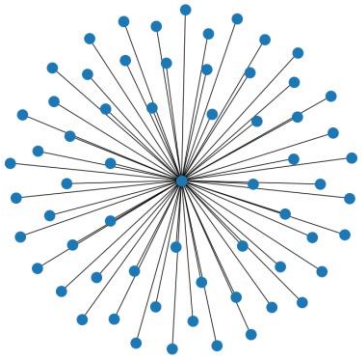


Problem Statement 3/3

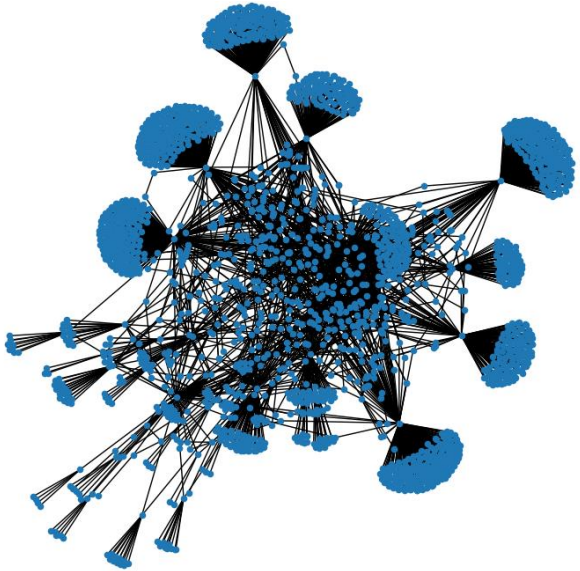
NTT Security Holdings

- In an Internet-scale network, the number of neighbors of a given host can be very large.
- Therefore, it is essential to have an algorithm that can **efficiently** detect malicious hosts from the **huge graph**.

1-hop neighborhood

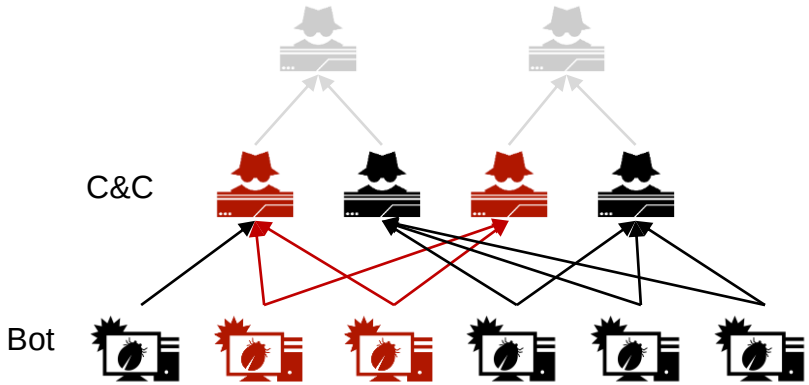


2-hop neighborhood



Basic Idea for Detection

- We propose a novel detection method, **BLC** (**B**lock**L**ist **C**o-occurrence analysis).
- To detect malicious servers on the IP graph, we use **detection by co-occurrence** [1].
  - Malicious actors might prepare not only 1 malicious server but several servers.
  - Infected hosts might connect to several malicious servers.
- For efficient detection on huge graph, we additionally use **pruning technique**.



[1] K. Sato, K. Ishibashi, T. Toyono, H. Hasegawa, and H. Yoshino, ‘‘Extending black domain name list by using co-occurrence relation between DNS queries,’’ IEICE Trans. Commun., vol. E95.B, no. 3, pp. 794–802, Mar. 2012.

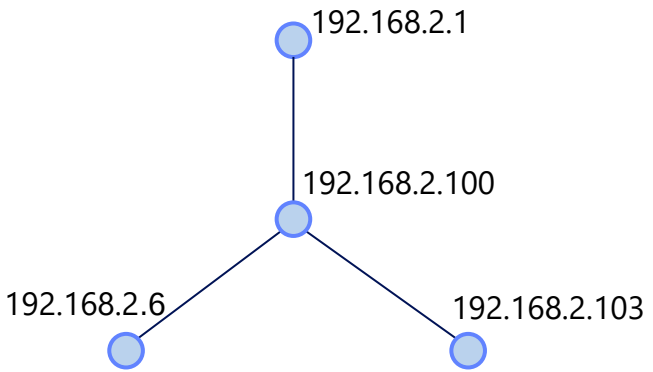
Proposal: Graph Construction

- We generate a graph of communication relationships between IP addresses from flow data.
- The graph is undirected, because the sampled flow data do not always tell us which host initiated the communication.

Flow

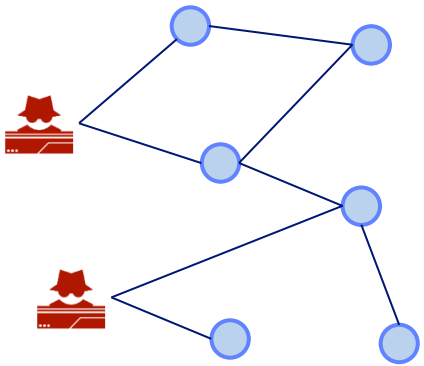
SrcIP	DstIP	...
192.168.2.1	192.168.2.100	
192.168.2.6	192.168.2.100	
192.168.2.100	192.168.2.1	
192.168.2.103	192.168.2.100	
:	:	

Graph

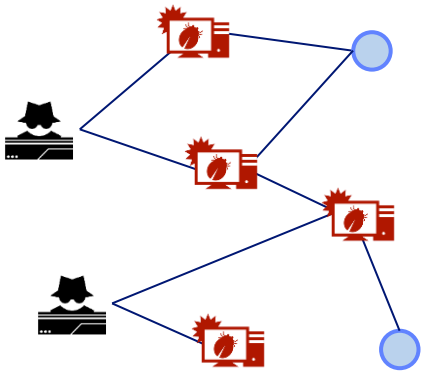


Proposal: Listing Up Candidates

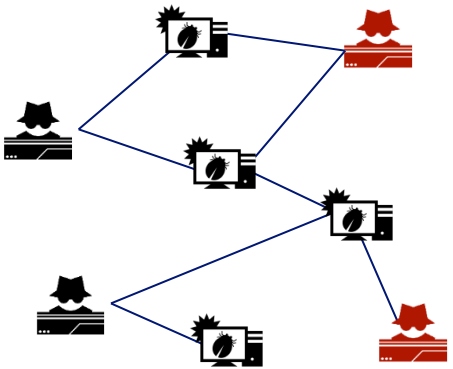
- We assume all IPs communicating with blocklist IPs are bots.
- IPs communicating with bots that are neither bots nor blocklist IPs are considered malicious IP candidates.



Seeds: Blocklist



1-hop: Bot Candidates

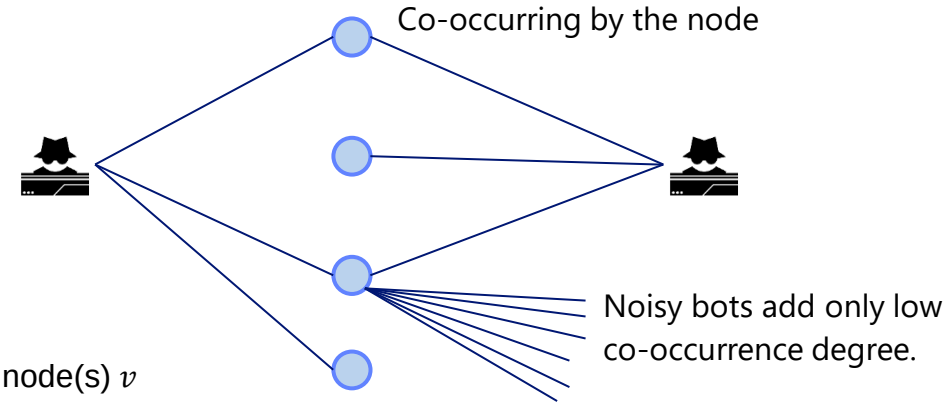


2-hop: C&C Candidates

Proposal: Score Calculation 1/2

- Co-occurrence between hosts is calculated by the similarity of the communication destination set.
- To reduce the effect of noisy nodes such as scanners, it is calculated as a **weighted Jaccard Index** of neighborhood as follows:

$$C(h_i, h_j) = \frac{\sum_{v \in N(h_i) \cap N(h_j)} 1/|N(v)|}{|N(h_i) \cup N(h_j)|}$$



※ $N(v)$: neighborhood of node(s) v
h_i : i-th host

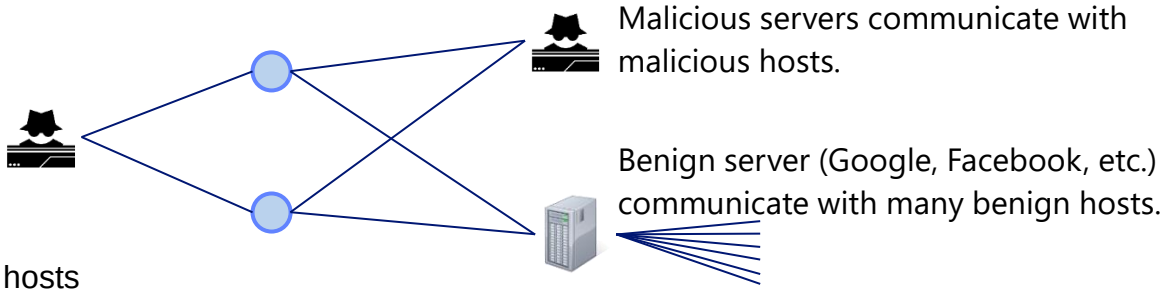
Proposal: Score Calculation 2/2

- Since bots often communicate with benign services to check for connection, we consider malicious weight by **rate of bots in neighborhood**:

$$\mathrm{W}(h_i) = \frac{|N(H_{mal}) \cap N(h_i)|}{|N(h_i)|}$$

- The final malicious score is **product of co-occurrence and malicious weight**:

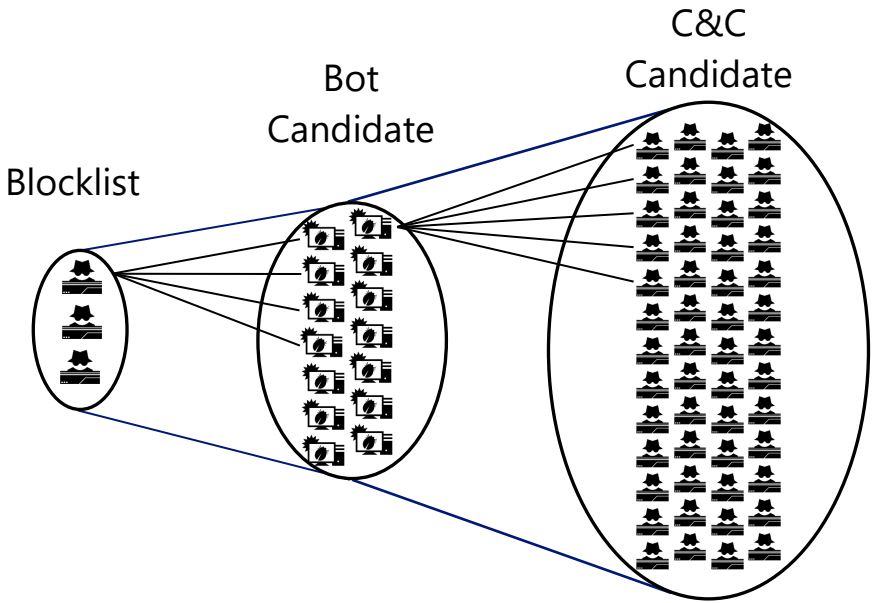
$$\mathrm{Mal}(h_i) = \mathrm{W}(h_i) \times \sum_{h \in H_{mal}} \mathrm{C}(h, h_i)$$



※ H_{mal} : Set of blocklist hosts

Proposal: Pruning Technique 1/2

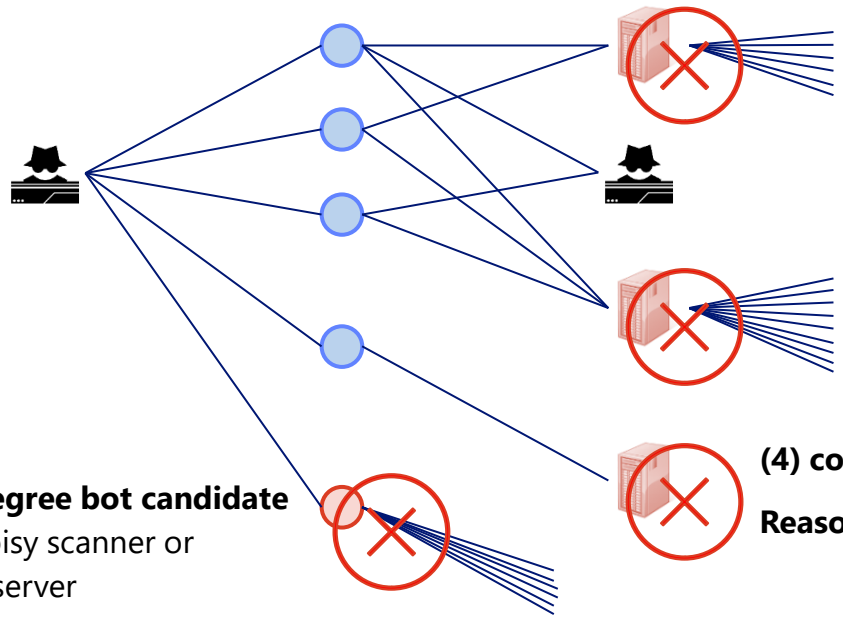
- To calculate the scores for all C&C candidates, we need the weighted Jaccard index calculation of (# block list IPs) x (#C&C candidate IPs) times.
- This is a very time-consuming operation because the size of a 2-hop neighborhood can be very large in an Internet-scale graph.



Proposal: Pruning Technique 2/2

NTT Security Holdings

- The following pruning heuristic is applied.



(1) high degree bot candidate
Reason: noisy scanner or legitimate server

(2) small malicious weight
Reason: legitimate server

(3) communicating to too many hosts
Reason: unlikely to be C&C

(4) communicating to only a few bot
Reason: insufficient evidence

Evaluation: Dataset and Parameters

NTT Security Holdings

- Flow data: Real flow data of a large network
- Blocklist: Seed blocklist (general C&C, IoT C&C)

Statistics of data:

Item	**Size**
Flow records (per day)	2.2×10^9
Graph nodes (per day)	1.8×10^8
Graph edges (per day)	6.3×10^8
General C&C blocklist	3932
IoT C&C blocklist	483

proposed pruning parameter:

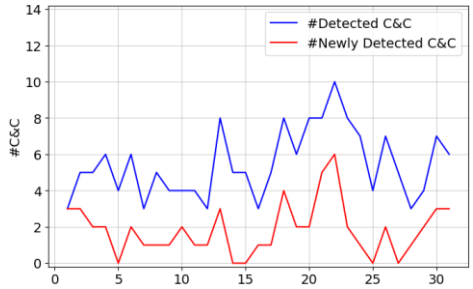
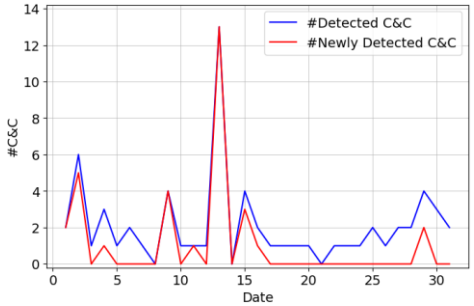
Parameter	**Meaning**
bot_deg = 100	degree of bot candidate > **bot_deg**
weight = 0.1	malicious weight > **weight**
c2_deg = 1000	degree of C&C candidate < **c2_deg**
comm_bot = 4	# of communicating bots > **comm_bot**

Evaluation: Validation with Analyst 1/2

NTT Security Holdings

The result of the **proposed method** (BLC)

- We extract 100 most suspicious host per day for 1 month.
- The result is validated by a security analyst using OSINT, which is independent of the seed block list.

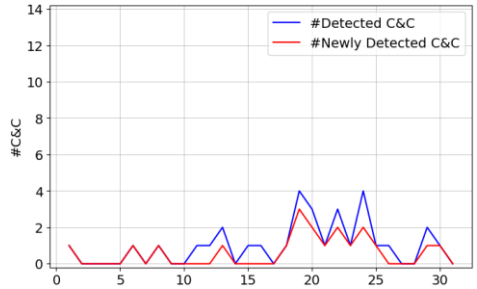
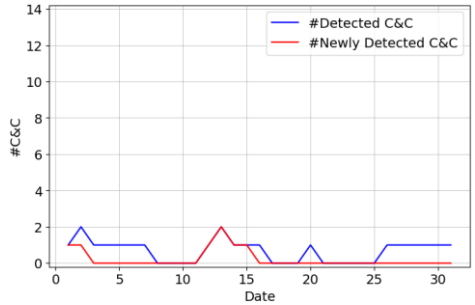
Used blocklist	#detected C&C server	#C&C / #Unique IPs	Percentage expansion of block list
General C&C blocklist		57 / 909	57 / 3932 = 1.4%
IoT C&C blocklist		32 / 862	32 / 483 = 6.6%

Evaluation: Validation with Analyst 2/2

NTT Security Holdings

The result of the **conventional method** (BLC without pruning)

- It **detects fewer C&C servers** than the proposed method.

Used blocklist	#detected C&C server	#C&C / #Unique IPs	Percentage expansion of block list
General C&C blocklist		19 / 1382	19 / 3932 = 0.5%
IoT C&C blocklist		7 / 859	7 / 483 = 1.5%

Evaluation: Effect of Pruning

NTT Security Holdings

Change in computation time with pruning

- Proposed pruning significantly reduces computation time.
- The score calculation is parallelized on 32 cores.

Method	**#2-hop nodes**	**Process time (minutes)**	**Similarity※ of top 100 IPs with conventional method**
Conventional method [1] (BLC without pruning)	5.4×10^7	161.4	100%
BLC (bot_deg=3000, weight=0.1)	8.9×10^6	26.6	98%
BLC (proposed pruning)	1.4×10^5	**0.7**	28%

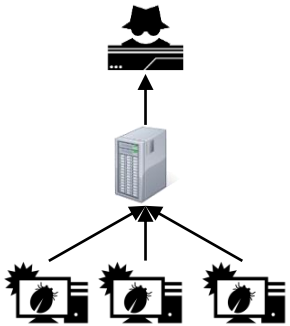
※by Jaccard index

[1] K. Sato, K. Ishibashi, T. Toyono, H. Hasegawa, and H. Yoshino, ‘‘Extending black domain name list by using co-occurrence relation between DNS queries,’’ IEICE Trans. Commun., vol. E95.B, no. 3, pp. 794–802, Mar. 2012.

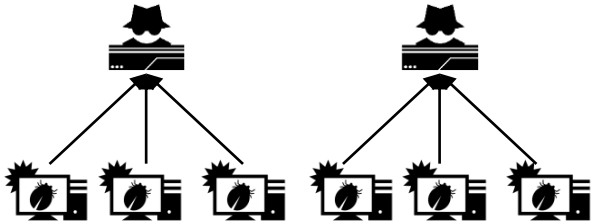
Limitation

- The proposed method assumes that bots communicate directly with multiple C&C servers. Therefore, It is not effective in the following cases:

C&C and bots communicate through proxy servers.



Each bot communicates with only one C&C. (C&C servers are not connected.)



Conclusion

NTT Security Holdings

- We propose a method (BLC) to detect malicious hosts related to the given blocklist hosts from flow data.
- It works efficiently even for Internet-scale IP network traffic flows and is mote than 100 times faster and higher precision than conventional method.
- Evaluation using large real flow data show that BLC find many C&C servers.

Appendix: Malware Types

NTT Security Holdings

- BLC detects hosts that are the C&C servers for the following malwares.

from general C&C blocklist

Malname	Type	#IP
win.socks5_systemz	Other	35
elf.mirai	DDoS	6
elf.mozi	DDoS	3
unknown.unknown	Other	3
win.asyncrat	RAT	1
script.coinMiner	Unknown	1
elf.bashlite	DDoS	1
win.rhadamanthys	CredentialStealer	1
win.teamspy	RAT	1
8220-Gang	Unknown	1
win.cobalt_strike	PentestFramework	1
elf.gafgyt	DDoS	1
dll	Unknown	1
elf.unknown	Other	1

from IoT C&C blocklist

Malname	Type	#IP
elf.mirai	DDoS	8
elf.bashlite	DDoS	4
win.cobalt_strike	PentestFramework	3
script.unknown	Other	2
win.dcrat	RAT	2
win.redline_stealer	CredentialStealer	1
win.nanocore	RAT	1
win.quasar_rat	RAT	1
elf.unknown	Other	1
win.shadowpad	Backdoor	1
win.icedid_downloader	CredentialStealer	1
win.sliver	PentestFramework	1
ascii.unknown	Other	1
elf.opendir	Other	1
win.bazarbackdoor	Backdoor	1
script.coinMiner	Unknown	1
win.icedid	CredentialStealer	1