Recommended Paper

d-ACTM/VT: A Distributed Virtual AC Tree Detection Method

Nobutaka Kawaguchi,[†1] Hiroshi Shigeno[†1] and Ken-ichi Okada[†1]

In this paper, we propose d-ACTM/VT, a network-based worm detection method that effectively detects hit-list worms using distributed virtual AC tree detection. To detect a kind of hit-list worms named Silent worms in a distributed manner, d-ACTM was proposed. d-ACTM detects the existence of worms by detecting tree structures composed of infection connections as edges. Some undetected infection connections, however, can divide the tree structures into small trees and degrade the detection performance. To address this problem, d-ACTM/VT aggregates the divided trees as a tree named Virtual AC tree in a distributed manner and utilizes the tree size for detection. Simulation result shows d-ACTM/VT reduces the number of infected hosts before detection by 20% compared to d-ACTM.

1. Introduction

Most existing network worms such as Blaster, Codered and Slammer conduct random address scans to find vulnerable hosts. Recently, however, the emergence of the hit-list worm has been investigated[14),15)]. The hit-list worm is a network worm that has a complete list of vulnerable hosts (hit-list) and propagates rapidly using the list. Most existing detection methods cannot detect the hit-list worms effectively.

To detect a kind of hit-list worms named Silent worms in a distributed manner, d-ACTM (Distributed Anomaly Connection Tree Method) was proposed[6),9)]. d-ACTM focuses on two features of the propagation activities of most network worms. The first is that worms propagation activities are expressed as tree-like structures. The second is that when selecting infection targets, the worms do not care about to which hosts and how frequently its infected host has communicated. Therefore, d-ACTM detects the worms by detecting tree structures consisting of Anomaly Connections (AC).

A problem with d-ACTM is that some infection connections classified as normal connections can divide the tree structures into small trees, and then the detection performance can be degraded.

To address the problem, in this paper we propose a network-based worm detection method named d-ACTM/VT (d-ACTM with Virtual AC Tree Detection). d-ACTM/VT aggregates such divided small trees into a tree named Virtual AC tree (VAC Tree) in a distributed manner, and detects the existence of worms when the tree size exceeds the threshold. In d-ACTM/VT, messages from IDSes which monitor the root hosts of AC trees are transmitted through Internal Connections (ICs) to IDSes which monitors the root hosts of AC trees located at upper side of sender side AC trees across the ICs. Here, IC is a connection between two internal hosts in a network.

Although a worm detection method named ACTM also has the ability to detect VAC trees[8)], it requires a central detection server that obtains global knowledge of the network and analyzes connection logs of all internal hosts, and therefore is not scalable compared to d-ACTM[6),9)].

The following sections are organized as follows. In Section 2, we will explain the Silent worm and d-ACTM. We state the problem with d-ACTM and then propose d-ACTM/VT in Section 3. In Section 4, the performance of d-ACTM/VT is evaluated. We introduce existing worm detection methods in Section 5. Finally, Section 6 concludes this paper.

2. Silent Worm and d-ACTM

In this section, we describe the model of Silent worm and d-ACTM[6),9)].

2.1 Model of Silent Worms

Silent worm is modeled as follows[6),9)].

(1) The worm has a complete hit-list of vul-

†1 Faculty of Science and Technology, Keio University

The initial version of this paper was presented at the 130th SIG Conference of DPS held on March 2007 sponsered by SIGDPS and SIGCSEC, whose proceedings are IPSJ SIG Technical Reports 2007-DPS-130. This paper was recommended to be submitted to IPSJ Journal by the chariman of SIGDPS.

nerable hosts in the target network. Each copy randomly selects infection targets from the list without any duplicated infection trials for the same hosts.

(2) The worm exploits the vulnerabilities of network services of both clients/servers.

(3) The number of infection trials of each worm copy is limited to a few times.

(4) The worm infects vulnerable hosts via unicast connections.

Worms can avoid duplicated trials by dividing the hit-list each time a new host is infected[15].

2.2 Connection Model

In d-ACTM, a connection between two internal hosts of a network is referred to as an Internal Connection (IC). The ICs include both the TCP connections and UDP flows between internal hosts. We assume d-ACTM/VT can identify each TCP connection and each UDP flow from network traffic. The ICs consist of the Legitimate Connections (LCs) and the Worm Connections (WCs); LCs are ICs created through a legitimate network activity while WCs are ICs created by the worms infection activity. Note that, d-ACTM cannot identify WC directly since its detection target is zero-day worm and no worm signature is available.

The ICs can be classified based on their occurrence ratios. Here, the occurrence ratio of an IC means a ratio of the number of the IC's occurrence in a period to the number of all ICs its source host has opened in the period. ICs opened with higher occurrence ratio than a certain threshold are classified into Normal Connections (NCs) and others are classified into Anomaly Connections (ACs).

2.3 Overview of d-ACTM

d-ACTM detects the existence of worms through the distributed detection of the tree structures composed of ACs as edges. The tree is called an AC tree. We define the size of an AC Tree as the number of hosts included in the tree. **Figure 1**(a) shows an example of AC trees. In this Figure, two trees rooted at host A and host C are depicted, and the size of the trees are 5 and 4 respectively.

Figure 1(b) shows the overview of d-ACTM. d-ACTM consists of IDSes named Local Anomaly Connection Detectors (LACDs). Each LACD is responsible for monitoring the inbound and outbound ICs of its target host. Then, by exchanging the monitoring results to each others, LACDs cooperatively detect AC trees.

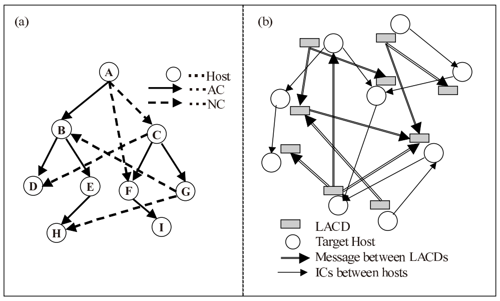


Fig. 1 (a) AC tree (b) Overview of d-ACTM.

d-ACTM exploits the following features of the worm propagation activities for detection.

(1) The worms infection activity constructs a tree structure composed of the infected hosts as nodes and the WCs as edges.

(2) When selecting the infection targets, the worm does not care to which hosts and how frequently the current host communicated before the infection.

On the other hand, benign hosts tend to communicate to a small portion of hosts in the network frequently[1),11)]. Most WCs are classified as ACs, and the most of LCs are classified as NCs. Therefore, when an AC tree is detected and the size of the tree exceeds a threshold, d-ACTM detects the existence of worms.

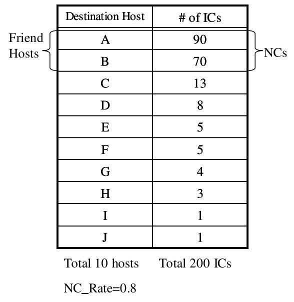
For the detection, LACD (1) classifies ICs of its target host, (2) estimates the size of AC trees, (3) and adjusts its detection threshold.

2.4 IC Classification

Each LACD periodically classifies its target host's outbound ICs into either NCs or ACs based on logs of outbound ICs monitored for a past certain interval. For example, assume the LACD conducts classification in every Tp, the current time is Tc, and the length of the interval is Ti. In this case, logs monitored from $Tc-Ti$ to Tc are analyzed at Tc, and then logs monitored from $Tc+Tp-Ti$ to $Tc+Tp$ are analyzed at $Tc+Tp$.

In each classification analysis, as shown in **Fig. 2**, LACD sorts the tuples of hosts and the number of outbound ICs of its target in an interval according to the number of ICs in descending order.

Next, LACD adds hosts corresponding to the top NC_Rate ($0 \leq NC\_Rate \leq 1$) ICs in the sorted list to a list named Friend Host List. After that, ICs destined for hosts in the Friend Host List are classified as NCs and other ICs are classified as ACs until the next analysis. Here, NC_Rate is a parameter which specifies



Destination Host	# of ICs
A	90
B	70
C	13
D	8
E	5
F	5
G	4
H	3
I	1
J	1

Friend Hosts
NCs
Total 10 hosts
Total 200 ICs
NC_Rate=0.8

Fig. 2 Sorted list of ICs.

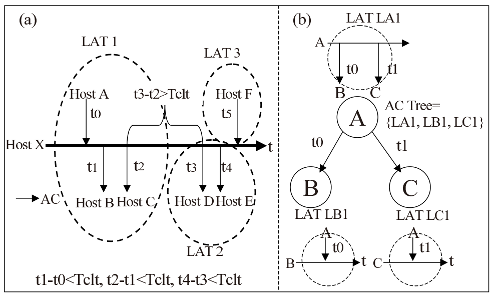


Fig. 3 Local AC concatenation.

the ratio of outbound ICs classified as NCs to the total number of outbound ICs.

In Fig. 2, the total number of ICs in the sorted list is 200, and NC_Rate is set to 0.8. Thus 160 ICs are classified as NCs. Then hosts A and B are added to the Friend Host List. As a result, ICs destined for host A and B are classified as NCs, and the other ICs are classified as ACs.

On the other hand, LACD cannot classify inbound ICs directly. Thus, when a host opens an outbound NC, its LACD notifies the occurrence to the receiver-side LACD via NC Notification Message. The message indicates that ICs from the sender side host to the receiver side host should be classified as NCs. Using the message, the receiver side classifies inbound ICs into either NCs or ACs.

2.5 AC Tree Detection

2.5.1 Local AC Concatenation

To manage ACs monitored at a target host, the LACD concatenates inbound and outbound ACs, the generation times of which are close to each other, as a group named LAT (Local AC Trees). **Figure 3** (a) shows an example of LATs at host X.

There are two cases where a new LAT is generated. First, when an LACD detects an inbound AC, a new LAT with the AC is always generated. In Fig. 3 (a), LAT1 and 3 are generated when inbound ACs from Host A and F are monitored respectively. The LACD judges whether an inbound IC is an AC or NC based on the NC Notification Messages.

Second, when LACD detects an outbound AC, a new LAT is generated if no AC has been monitored for more than a certain time period. We name the period *CLT* (Connection Limit Time). CLT is a parameter of LACD that controls the generation of new LAT. In Fig. 3 (a), CLT is set to T_{clt} and when an outbound AC is monitored at t_3, LAT2 is generated since $t_3 - t_2 > T_{clt}$.

On the other hand, if there exist LATs which include ACs detected within past CLT from the detection time of a new outbound AC, the AC is concatenated with the LATs. In Fig. 3 (a), ACs monitored at t_1 and t_2 are concatenated with LAT1 and an AC at t_4 is concatenated with LAT2.

A worm can be detected fastest when the interval of infection connections of the worm is equal to CLT.

Here, a LAT is a part of an AC tree. Thus, multiple LATs at different LACDs will correspond to one AC tree. In Fig. 3 (b), three LATs LA1, LB1 and LC1 are at different LACDs, and correspond to an AC Tree. Each LACD maintains the status of an AC tree, in which its target host is included, as a tuple named ACT_Info. Each ACT_Info consists of ACs that compose a LAT and the size of an AC tree, to which the LAT corresponds. We name the size S_{ACT}. The initial value of S_{ACT} is 1, and as a new AC is added to a LAT, the corresponding S_{ACT} is increased by 1. Note that, each LACD computes S_{ACT} of an AC tree based on its LAT and information from other LACDs, as shown in the next section, and therefore S_{ACT} may differ from the actual size of the AC tree.

An ACT_Info is removed when its S_{ACT} has not been increased for a certain time.

2.5.2 Tree Size Estimation

To estimate the size of an AC tree, an LACD sends a message named AUM (ACT Update Message) each time S_{ACT} of an ACT_Info is increased by more than a certain value named TH_{update}. TH_{update} is a parameter of LACD to control the transmission of AUM.

The receiver of AUM is an LACD that monitors a *parent host* of sender LACD's target host in an AC tree corresponding to the ACT_Info. Here, the *parent host* of host X is a host that is located at the parent side of host X in an AC tree. For example, in Fig. 3 (b), host A is a parent host of host B and C.

AUM contains two data: (1) increased value of S_{ACT} since the last AUM for the corresponding AC tree is sent to the receiver and (2) information about an AC that links the sender and receiver LACDs. The receiver LACD finds the LAT that includes the AC specified by the AUM, and adds the increased value to the S_{ACT} of corresponding ACT_Info.

Then, if S_{ACT} at a receiver LACD exceeds the detection threshold named TH_{ac}, the LACD sends an alert to the Network Manager. Note that each LACD has its own TH_{ac} value, and thus TH_{ac}s of different LACDs may take different values.

Finally, if the target host of the receiver LACD has a parent host, the AUM is relayed to the LACD of the parent host recursively. Thus, the LACD of a root host receives AUMs from LACDs of all hosts in an AC tree rooted at its target host, and therefore is able to compute S_{ACT} that matches the actual size of the AC tree.

2.6 Threshold Adjustment

The Network Manager is a system that investigates the network and hosts to verify alerts received from LACDs. If an alert is a false alert, it returns the notification to the sender LACD.

Based on the notifications and a parameter named *DAI* (desirable false alert interval), each LACD adjusts its TH_{ac}. DAI specifies the desirable interval between two continuous false alerts generated by one LACD. When an LACD generates a false alert, TH_{ac} of the LACD is increased by a certain value named TH_{INC}, which is a parameter of LACD. On the other hand, if an LACD has not generated false alerts for $\frac{DAI}{TH_{INC}}$, its TH_{AC} is decreased by 1. With this approach, TH_{ac} of each LACD is being adjusted to a value that satisfies DAI. When the false alert interval of each LACD becomes DAI and there are N hosts in a network, 1 false alert is raised in every $\frac{DAI}{N}$ in the network.

3. d-ACTM/VT

In this section, we propose d-ACTM/VT (d-ACTM with distributed Virtual AC Tree Detection) that extends the current d-ACTM by introducing the distributed VAC tree detection.

3.1 Tree Division Problem

A problem with d-ACTM is that an AC tree caused by worms propagation can be divided into small trees by WCs classified as NCs. This will lead to the degradation of the detection performance. We call the problem a Tree Division Problem.

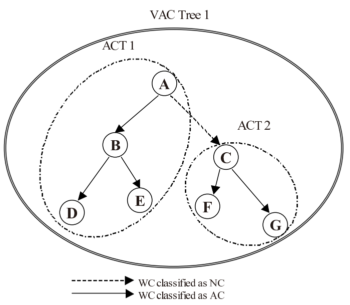


Fig. 4 Tree division problem.

Figure 4 shows a notion of the Tree Division Problem. A WC from host A to C is classified as an NC, and the WC divides the worms infection tree into two small AC trees (ACT1 and ACT2). Since d-ACTM generates an alert when the size of an AC tree exceeds TH_{ac}, such tree division delays the detection with d-ACTM, and therefore more hosts are infected before detection.

d-ACTM/VT addresses the Tree Division Problem through the *aggregation of AC trees* in a distributed manner. Since most WCs are classified as ACs, when an AC tree caused by worms propagation is divided by WCs classified as NCs, the distance between the divided sub trees will be short. Here, the distance is expressed as the minimum number of NCs that link the trees. In addition, most of the trees will grow up to a certain size since worms continue propagation regardless of the tree division. Thus, d-ACTM/VT aggregates trees that are larger than a certain size and close to each other from the viewpoints of distance and generation time as a group named Virtual AC Tree (VAC Tree). Then, when the tree size exceeds a threshold, d-ACTM/VT detects the worms. In Fig. 4, ACT1 and ACT2, the distance between which is 1, are aggregated into VAC Tree1.

In d-ACTM/VT, messages from the LACDs that monitor the root hosts of AC trees, are transmitted through ICs to the upper side AC trees to aggregate the information of trees.

Although ACTM[8] also has the ability to aggregate divided trees, it requires a central detection server and global knowledge of the network for the aggregation, and therefore is not scalable compared to d-ACTM[9]. Thus, we attempt to realize a method that achieves scalable VAC tree detection.

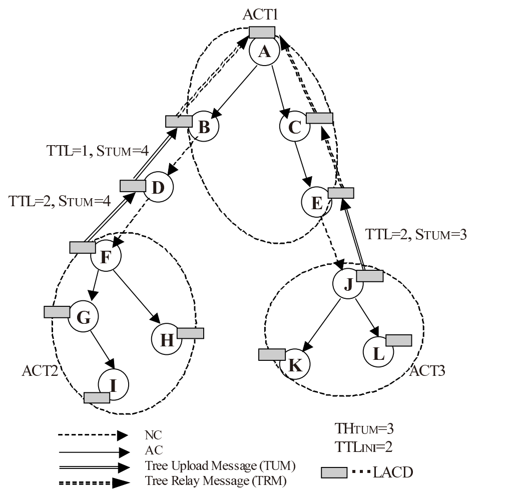


Fig. 5 Approach of distributed AC tree aggregation.

3.2 Tree Aggregation Approach

Figure 5 shows the approach of distributed AC tree aggregation. In Fig. 5, ACT2 and ACT3 are aggregated with ACT1 as a VAC tree.

When the size of an AC tree becomes equal to or greater than a threshold named TH_{TUM}, the LACD that monitors the root host of the tree sends TUM (Tree Upload message) to the LACD that monitors host which has opened NCs to sender's target host within a certain past time.

TUM includes (1) the size of an AC tree rooted at the target host, (2) an NC that links the sender and receiver LACDs, and (3) TTL. We denote (1) as S_{TUM}. TTL is a parameter of LACD that specifies the range of destinations of TUM. Initially, TTL is set to a value named TTL_{INI}.

As a TUM is received by an LACD, its TTL is decreased by 1. Then, if the TTL is still greater than 0, the receiver recursively transmits the TUM. In Fig. 5, $TTL_{INI} = 2$ and a TUM is transmitted from the LACDs of host F to B by way of host D.

When an LACD receives a TUM and its target host is included in an AC tree as a leaf or internal node, the LACD sends a new message named TRM (Tree Relay Message) to the LACD of the parent host in the AC tree. TRM includes (1) a TUM and (2) an AC that links the sender and receiver LACDs. The TRM is relayed to the root of the tree recursively. In Fig. 5, the LACD of host E receives a TUM from the LACD of host J, and sends a TRM, which contains the TRM, to the LACD of host C. Finally, the TRM is relayed to the LACD of host A, which is a root of ACT1.

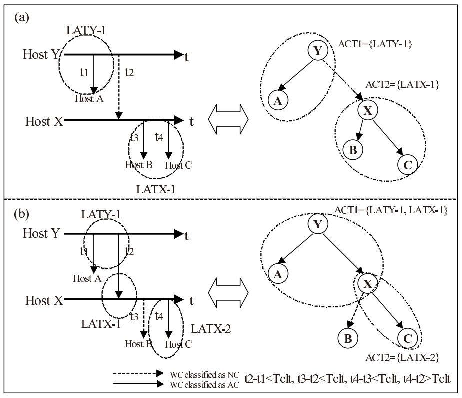


Fig. 6 (a) Remote LAT division, (b) Local LAT division.

When an LACD receives a TUM or TRM and its target host is a root of an AC tree T, the TUM or TRM is stored in the LACD. Then, the sum of the size of T and S_{TUM}s of the TUMs and TRMs is used as the size of a VAC tree. If the size exceeds a threshold named TH_{vac}, the LACD detects the existence of worms and sends an alert to the Network Manager. In Fig. 5, at the LACD of host A, the sum of S_{TUM}s in TRMs from host B and C are 7 (= 4+3), and the size of ACT1 is 4. Thus, the size of the VAC tree consisted of ACT1-3 is 11.

In the following sections, more detailed algorithms in LAT level will be given.

3.3 Division in LAT Level

As shown in Fig. 3 (b), in d-ACTM/VT, an AC tree is composed of some LATs, which are managed by different LACDs. In other words, each LAT corresponds to an AC tree. Thus, in LAT level, when tree division occurs, LATs that should correspond to one tree happen to correspond to different small trees.

There are two types of LAT division; (1) Remote LAT division and (2) Local LAT division as **Fig. 6** shows.

In remote LAT division, LATs at different hosts happen to correspond to different AC trees. In Fig. 6 (a), due to the tree division by a WC at t2, LATY-1 and LATX-1 correspond to different trees ACT1 and ACT2.

In local LAT division, LATs at the same host happen to correspond to different AC trees due to outbound WCs classified as NCs. In Fig. 6 (b), since a WC at t3 is classified as a NC and $t4 - t2 > T_{clt}$, LATX-1 and LATX-2 correspond to different trees, ACT1 and ACT2

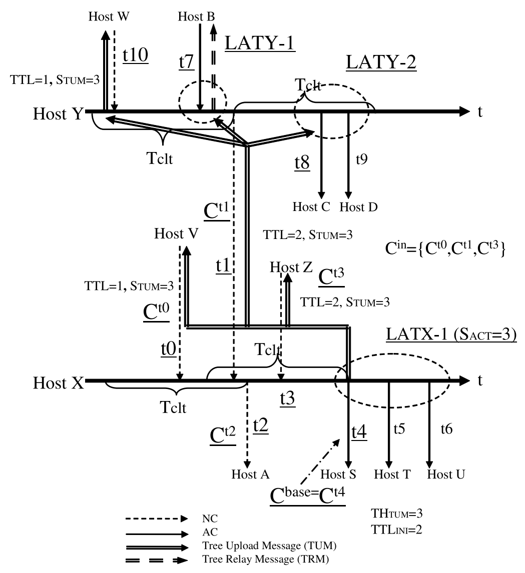


Fig. 7 Remote LAT aggregation.

respectively.

Thus, there are two types of LAT aggregation; Remote LAT Aggregation and Local LAT Aggregation.

3.4 Remote LAT Aggregation

3.4.1 Sender Side of TUM

Figure 7 shows the transmission of TUMs for the aggregation of LATs in different hosts.

LACD sends TUMs when S_{ACT} corresponding to an LAT becomes equal or greater than TH_{TUM}, and the LAT is a *Root LAT*. Here, Root LAT is a LAT that only contains outbound ACs. In Fig. 7, with ACs at t4, t5 and t6, S_{ACT} of LATX-1 becomes 3, which is equal to TH_{TUM}. Thus, TUMs are to be transmitted.

Here, R denotes a root LAT, S_{ACT} of which is equal or greater than TH_{TUM}. In Fig. 7, LATX-1 is R. Also, C_L^{first} and C_L^{last} denote the first and last AC of a LAT L respectively. C^T denotes an IC generated at time T. Also, a function $G(C)$ returns the generation time of an IC C.

The destination of a TUM is specified based on an IC named C^{base}. At first, C^{base} is set to the C_R^{first}. In Fig. 7, C^{base} is C^{t4}.

TUM is sent to the LACD monitoring a host which is a sender of an NC C^{in} that satisfies both the conditions.

(1) The destination of C^{in} is the sender host of C^{base}.

(2) C^{in} is linked with C^{base} by a sequence of ICs named *IC_Chain*, and the size of the IC_Chain is smaller than TTL.

The TTL of a TUM is decreased by the size of the IC_Chain before transmission.

Here, an IC_Chain is a sequence of ICs that consists of two arbitrary ICs as the first and last elements and the minimal number of outbound NCs generated between the ICs. In the IC_Chain, intervals between any adjacent elements must be not longer than CLT. The size of an IC_Chain is the number of outbound NCs within the chain. When two arbitrary ICs belong to the same IC_Chain as the first and last elements, we can say that the ICs are *linked* by the chain.

In Fig. 7, the sender host of C^{base} is Host X. Then, C^{t0}, C^{t1}, C^{t3} satisfy the condition (1). In the case of C^{t1}, C^{t1} is directly linked with C^{t4} since $t4 - t1 \leq T_{clt}$, and therefore C^{t1} is a C^{in}. Then, TUM with TTL $= 2$ is sent to the LACD that monitors the sender of C^{t1}, Host Y. The same analysis holds for C^{t3}.

In the case of C^{t0}, C^{t0} is linked with C^{t4} by an outbound NC C^{t2} since $t4 - t2 \leq T_{clt}$ *and* $t2 - t0 \leq T_{clt}$, and therefore satisfies the condition (2). Then, a TUM with TTL $= 1$ is sent to the LACD of host V, the sender of C^{t0}.

3.4.2 Receiver Side of TUM

On receiving a TUM, the receiver LACD decreases TTL of the TUM by 1. Then, the LACD associates a NC C^{in} in the TUM with LAT L that satisfies one of the following conditions.

(1) $G(C_L^{first}) \leq G(C^{in}) \leq G(C_L^{last})$

(2) L is a root LAT and a chain that links C^{in} and either of C_L^{first} or C_L^{last} exists.

(3) L is not a root LAT and a chain that links C_L^{last} and C^{in} $(G(C_L^{last}) \leq G(C^{in}))$ exists.

In the case either of (2) or (3), the chain size must be not greater than TTL of the TUM.

In Fig. 7, host Y is a sender of C^{t1}, which is one of C^{in}s. Then, LATY-1 satisfies (3) since $t1 - C_{LATY-1}^{last} \leq T_{clt}$. Thus, since LATY-1 has an inbound AC C^{t7}, a TRM that includes the TUM is transmitted to the LACD of host B, a parent host of host Y.

On the other hand, LATY-2 satisfies (2) since $C_{LATY-2}^{first} - t1 \leq T_{clt}$. The TUM is stored in the ACT_Info corresponding to LATY-2.

At the root LAT, the size of the VAC tree is computed. In Fig. 7, the size at LATY-2 is 5 $(= 2 + 3)$. Then, if the size exceeds TH_{vac}, an alert is generated.

Thus, the aggregation of LATX-1 and LATY-

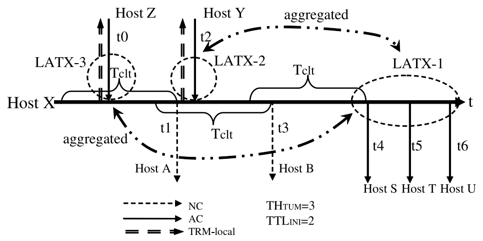


Fig. 8 Local LAT aggregation.

1, and aggregation of LATX-1 and LATY-2 are generated.

The LACD of host X sends TUMs each time S_{ACT} of LATX-1 is increased. Thus, when an LACD receives a new TUM corresponding to LATX-1, the LACD removes old TUM corresponding to LATX-1.

Next, if the TTL is still greater than 0, the TUM is transmitted recursively. In this case, C^{base} is set to C^{in}. In Fig. 7, since $t1 - t10 \leq T_{clt}$, the TUM is transmitted to the LACD of host W recursively.

3.5 Local LAT Aggregation

When remote LAT aggregation occurs, local LAT aggregation also occurs at the same time as **Fig. 8** shows. In the local aggregation, root LAT R is aggregated with LAT L generated before R if there exists an *IC_Chain* that links C_R^{last} and C_L^{first}, and the size is not greater than TTL_{INI}.

In Fig. 8, LATX-1 and LATX-2 are linked by an outbound NC C^{t3}, and LATX-1 and LATX-3 are linked by C^{t1} and C^{t3}. Thus, the aggregation of LATX-1 and LATX-2, and the aggregation of LATX-1 and LATX-3 are generated. At LATX-2 and LATX-3, S_{ACT} of LATX-1 is treated like S_{TUM} and used for the computation of VAC trees.

Next, if L has an inbound AC, the LACD sends a message that includes S_{ACT} of R to the source of the AC. We name the message TRM-local. In Fig. 8, two TRM-locals are transmitted from LATX-2 and LATX-3 to LACDs of host Y and Z respectively. Then, as in the case of TRM, TRM-local is relayed to a root of an AC tree and used for the VAC tree detection.

3.6 Time Range of NC Logs

LACD need to preserve logs of NCs in a range for tree aggregation. When T_R^{first} denotes the generation time of the oldest root LAT stored in the LACD, T_L^{first} denotes the generation time of the oldest non-root LAT, T_R^{latest} denotes the generation time of the latest root LAT, and T_C^{latest} denotes the generation time of latest appended outbound AC, the range is roughly expressed as

$$[Min(T_R^{first} - CLT * (TTL_{INI} - 1), T_L^{first}), \\ T_C^{latest} + CLT * (TTL_{INI} - 1) \quad]$$

for outbound NCs, and

$$[T_R^{first} - CLT * TTL_{INI}, T_R^{latest}]$$

for inbound NCs.

If a host sends and receives many NCs, the size of logs in the range may exceed the storage capacity. In this case, some logs should be removed. In this paper, we assume LACDs have enough memory capacity, and the removal of logs in the range does not occur. While how to remove the logs without any detrimental effect is one of our future works, we consider that removal of logs in order of generation time will be effective.

3.7 Threshold Determination

Here we explain the determination of two thresholds TH_{vac} and TH_{TUM}.

First, the determination of TH_{vac} is the same as that of TH_{ac}. Since there are two detection thresholds (TH_{ac} and TH_{vac}), when DAI $= T_{DAI}$, both the interval of false alerts due to the VAC tree detection and AC tree detection should be $2 * T_{DAI}$. Then, both thresholds are to be adjusted to satisfy $2 * T_{DAI}$.

Second, TH_{TUM} is determined as

$$TH_{TUM} = TH_{ac} \times R_{TUM} \qquad (1)$$

where R_{TUM} is a parameter of LACD and $0 \leq R_{TUM} \leq 1$. The idea behind the expression is that the size of the divided sub tree will be increased to a certain ratio of TH_{ac} if the tree is caused by worms propagation. As TH_{TUM} increases, the number of transmitted TUMs and TRMs are decreased.

Thus, R_{TUM} and TTL_{INI} have a major effect on the VAC tree detection performance.

4. Evaluation

In this section, we will describe the computer simulation to evaluate the performance of d-ACTM/VT.

4.1 Simulation Model

In this simulation, we assume an enterprise network where all internal hosts have vulnerabilities exploited by Silent worms. **Table 1** shows the default parameters of the simulation. Here, TU denotes a time unit.

The details of the model are given in the following sections.

Table 1 Default simulation parameters.

# of hosts	500
Interval of outbound LCs (TU)	10
Ratio of hosts in Frequent Communication Hosts	0.24
Ratio of # of LCs to Frequent Communication Hosts	0.8
# of infection trials per copy	2
Infection interval (TU)	10
NC_Rate	0.8
TH_{update}	1
CLT (TU)	10
Initial value of TH_{ac}	10
Initial value of TH_{vac}	10
TTL_{INI}	1
R_{TUM}	0.35
DAI (TU)	$5 * 10^6$
TH_{INC}	5
Period of IC Classification (TU)	$5 * 10^4$
Interval of ICs analyzed at a time (TU)	$5 * 10^4$

4.1.1 Network Model

As a network model, we assume an enterprise network where common client-server type network services such as SSH, Windows RPC Services, Web, Mail are in operation, but P2P applications, which cause tree like connection structures, are not run. Actually, many organizations prohibit such P2P applications due to security reasons. The number of hosts is 500.

Patterns of the destination hosts of LCs in the network are modeled from 2 viewpoints. Model (1) focuses on how biased the destination hosts of LCs opened by each individual host are. Model (2) focuses on the bias of the destination hosts of LCs opened by all hosts in the network.

As for model (1), we classify the destination hosts of each individual host into 2 groups: Frequent Communication Hosts which include x% of all hosts in the network, and Infrequent Communication Hosts that include the other hosts. Then, 80% of LCs of each host are destined to its Frequent Communication Hosts and the others are for Infrequent Communication Hosts. Each host evenly opens LCs for the hosts in its Frequent Communication Hosts. The same holds for the hosts in its Infrequent Communication Hosts. As a default, x = 24%.

As for model (2), the members of Frequent Communication Hosts of a host are randomly selected from all hosts so that each host receives LCs from the other hosts with the same frequency from each other. Although the model may be somewhat different from typical networks with client-server services, this is a more difficult setting for worm detection sides since the kind of IDSes that focus on the occurrence of many connections destined from server to client hosts in the event of worms propagation[4] cannot be applied. Note that, however, the detection performance of d-ACTM/VT is almost the same in networks where hosts are either of the clients or server type hosts.

As for the open interval of outbound LCs, in order to evaluate the characteristic of d-ACTM/VT under a basic network model, we assume that the open frequencies of outbound LCs of all hosts are almost the same. Then, the interval between two continuous LCs opened by all hosts follows the exponential distribution and the average is 10 TU.

4.1.2 LACD Setting

We assume NC_Rate of all LACDs are set to 0.8, with which the detection performance of d-ACTM/VT against the destination bias model explained in Section 4.1.1 is optimized. Here, in reality, each host may have a different bias of destination hosts from each other, and the bias can be changed as time advances. Thus, it is desirable that each LACD can automatically adjust the NC_Rate to an optimized value based on the activity of its target host, and this is one of our future works.

Next, we assume that every LACD sets CLT to 10 TU. Therefore, d-ACTM/VT can detect worms with an infection interval shorter or equal to the average interval of outbound LCs. Since DAI is set to $5 * 10^6$ TU, the average interval of continuous false alerts raised in the network converges to 10^4 TU. If 1 TU = 1 sec, 1 false alert will be generated in about every 3 hours, which is considered to be a reasonable rate.

4.1.3 Silent Worm Model

As a worm model, we assume a Silent worm that has address lists of all vulnerable hosts in the network. The worm infects hosts by exploiting the vulnerabilities of the services run on the hosts. The number of infection trials per each infected host is limited to 2 at most to evade detection methods that focus on the connection rate[7]. We assume the infection interval of the worm is 10 TU, which is equal to the average interval of outbound LCs. Here, the infection interval is the average interval between the generation times of 2 continuous WCs opened by an infected host. The interval between when a host is infected and when the host starts infection activity is also 10 TU.

In this simulation, after the average of the

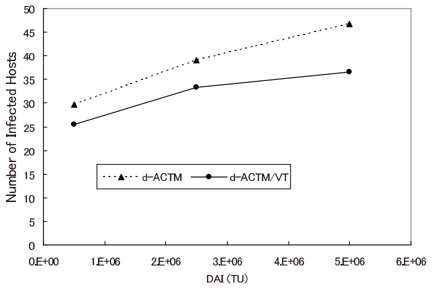


Fig. 9 The number of infected hosts before d-ACTM/VT detects worms.

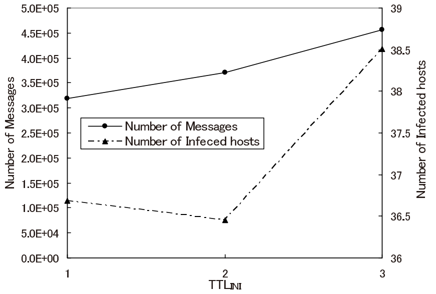


Fig. 11 The effect of TTL_{INI} on the detection performance.

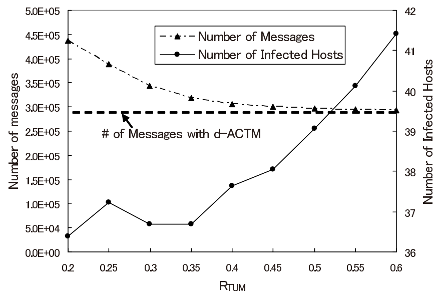


Fig. 10 The effect of R_{TUM} on the detection performance.

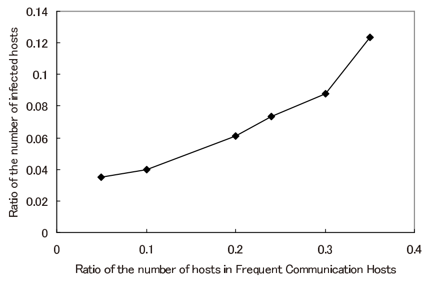


Fig. 12 The effect of the ratio of hosts in frequent communication hosts.

false alert interval in the network becomes 10^4 TU, a Silent worm infects one randomly selected host and starts propagation.

The following simulation results are the averages of 200 trials (Table 1).

4.2 Simulation Results

4.2.1 The Number of Infected Hosts

Figure 9 shows the number of infected hosts before detection with d-ACTM/VT and d-ACTM as a function of DAI.

As Fig. 9 shows, d-ACTM/VT can detect worms faster than d-ACTM. For example, with DAI $= 5*10^6\, TU$, d-ACTM/VT detects worms when 36 hosts are infected, and reduces the infected hosts by more than 20% compared to d-ACTM.

4.2.2 The Effect of R_{TUM}

Figure 10 shows the number of infected hosts and the average number of messages transmitted by each LACD as a function of R_{TUM}. The message contains TUMs, TRMs and other types of messages related to d-ACTM. As R_{TUM} increases, the number of transmitted messages is decreased, and the number of infected hosts is increased. With $R_{TUM} = 0.35$, the number of transmitted messages is increased by 9% compared to d-ACTM.

Thus, it can be said that d-ACTM/VT achieves a faster detection compared to d-ACTM with a small increase of transmitted messages.

4.2.3 The Effect of TTL_{INI}

Figure 11 shows the number of infected hosts and the average number of messages as a function of TTL_{INI}. As TTL_{INI} increases, the number of transmitted messages is increased. As to the number of infected hosts before detection, with $TTL_{INI} = 2$, the number is slightly reduced. With $TTL_{INI} = 3$, the number is steeply increased by 2. Thus, we consider $TTL_{INI} = 1$ achieves both fast detection and small network overhead.

4.2.4 The Effect of the Bias of the Destination Hosts of Outbound LCs

Figure 12 shows the ratio of the number of infected hosts as a function of the ratio of the number of hosts in Frequent Communication Hosts to the number of all hosts in the network. As the figure shows, when the ratio of hosts in Frequent Communication Hosts is smaller than 0.3, d-ACTM/VT can detect worms before 10% of hosts are infected.

Here, in many networks, the most hosts frequently connect to only a small percent of hosts in the network [1),11)]. Therefore, from the viewpoint of the bias of the destination hosts of LCs, d-ACTM/VT is effective in most networks.

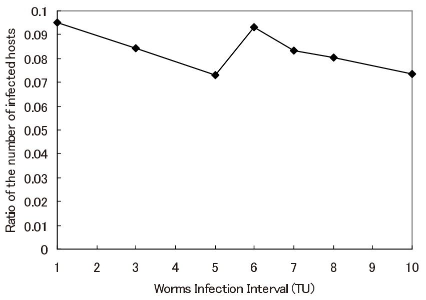


Fig. 13 The effect of worms infection interval.

4.2.5 The Effect of Infection Interval on the Detection Performance

Figure 13 shows the ratio of the number of infected hosts as a function of a worm's infection interval. From the figure, d-ACTM/VT can detect worms with infection interval $\leq$ 10 TU before 10% of all hosts are infected.

Here, the intervals of most existing worms is much shorter compared to the average interval of LCs[7]. Thus, from Fig. 12 and Fig. 13, d-ACTM/VT is effective against most ilent worms in most networks. In more detail, d-ACTM/VT can detect worms with infection interval $\leq$ 10 TU before 10% of hosts are infected in the network where 80% of outbound LCs of each host are destined to 24% of all hosts. Considering it is difficult for existing methods to detect Silent worms effectively, the detection performance of d-ACTM/VT is quite promising.

Also, Fig. 13 shows that as the infection interval decreases, the infected hosts are increased, except the point where the infection interval decreases from 6 TU to 5 TU. The reason is that when a WC is classified as NC, its previous and next WCs can be concatenated if the infection interval is equal or less than 5 TU.

4.2.6 Comparison with ACTM

ACTM, which requires a central server for detection, detects the existence of worms before 37 hosts are infected with a condition similar to Table 1. Thus, d-ACTM/VT can detect worms as fast as ACTM without any global knowledge of the network.

As to the network overhead, the number of IC logs transmitted to the ACTM server from several traffic capturing points in the network is 50 per 1 TU. With d-ACTM/VT, on the other hand, the number of messages transmitted among LACDs is only 4 per 1 TU.

Next, as to the computation cost for detection, the computation cost of the ACTM server is proportional to the number of hosts in the network. With d-ACTM/VT, on the other hand, each LACD only needs to analyze the part of AC trees in which its target host is included. This means, the computation cost of each LACD is scalable from the viewpoint of the number of hosts in the network. In addition, the ACTM server regards each AC tree as a group of sub trees like LATs, and therefore the analysis approach is similar to d-ACTM/VT. Thus, the computation cost of the ACTM server and the total costs of all LACDs are not so different except that, with d-ACTM/VT, the exchange of information of LATs among LACDs causes some network transmission costs. Therefore, the computation cost of each LACD is smaller compared to the ACTM server.

The same discussion holds for the data storage size required for the detection. The storage size of each LACD does not depend on the number of hosts in the network.

Thus, from the viewpoints of network overhead, computation cost and storage size, d-ACTM/VT is more scalable compared to ACTM.

4.2.7 Cost of VAC Tree Detection

The introduction of VAC tree detection involves some costs such as (1) the increase of the number of transmitted messages, (2) the increase of stored data due to inbound and outbound NC logs, (3) the extension of time for the false alert interval in the network to be stable as specified by DAI.

As for (1), the number of increased messages is small as mentioned in Section 4.2.3, and therefore the increase of the network overhead will not be a serious problem.

As for (2), since inbound and outbound NC logs are used for VAC tree detection, the storage size needed for detection is increased. In this simulation, the number of logs that need to be stored at a time is about 20. Since the size of each NC log is small, the required storage size for NC logs is small enough.

As for (3), since d-ACTM/VT uses two types of detection thresholds, it takes a longer time for each LACD to adjust its detection thresholds to optimal values compared to d-ACTM. Here, the optimal value is a value of threshold, with which the interval of continuous false alerts becomes almost equal to DAI. As a result, it takes a longer time for the false alert interval in the network to be stable as specified by DAI. In the simulation, with d-ACTM/VT, it takes $9.0 * 10^6$ TU before the average interval of false alerts in the network becomes 10^4 TU, and this is 1.4 times longer compared to d-ACTM. We consider, however, the time can be shortened through the improvement of the adjustment algorithms such as the exchange of thresholds values among LACDs. This will be one of our future works.

Therefore, the cost of the introduction of VAC tree detection is not serious or critical.

4.2.8 Discussion

In this simulation, we assume that the LC open interval of all hosts takes the same value. Here, in the case where some hosts open LCs with much higher frequency compared to the other hosts, detection thresholds of d-ACTM/VT are increased and then, more hosts will be infected before detection. We consider that one solution to address the problem is to exclude hosts with high frequent communication from the detection targets. How to identify and exclude such hosts in a distributed manner is one of our future works.

5. Related Works

Many existing network-based worm detection methods exploit the anomaly network behaviors of worms such as address scans[3),5),7)]. Since hosts infected by Silent worms do not conduct such anomaly activities, however, it is difficult to detect the worms with these approaches.

Xie, Y. has proposed a traceback method[18),20)] that identifies the origin infection host by tracing back communication logs in a network after the propagation is completed. d-ACTM/VT traces forward the connections occurred in the network as well, but its objective is not to identify the origin host but to detect the existence of worms in an early stage of propagation.

McDaniel, P., et al. proposed a worm containment method[10)] by blocking connections, which should not occur in normal operations. Also, some authors proposed methods that detect the worms by detecting trees[4),16)] or chains[17)] composed of connections. d-ACTM/VT focuses on the anomaly of each connection and then detects tree structures composed of the anomaly connections using a novel approach, and therefore is different from these methods.

Antonatos, S., et al. have proposed a method that changes hosts IP addresses frequently to decay the hit-list of worms[12)]. Although the method may be effective against hit-list worms, we consider the approach will be impractical in many network environments.

There are various approaches to detect worms in a distributed manner[2),13),19)]. Though these methods can detect traditional scanning worms effectively, however, they are ineffective against hit-list worms.

6. Conclusion

Tree Division Problem has bad effect on the detection performance of ACTM against Silent worms. To address this problem, in this paper, we have proposed d-ACTM/VT. By aggregating divided AC trees in a distributed manner and utilizing the tree size for detection, d-ACTM/VT detects the existence of worms.

Through the simulation experiments, we have shown that d-ACTM/VT detects worms before 10% of hosts are infected, and reduces the infected hosts by 20% with a small increase of network overhead compared to d-ACTM, and is more scalable compared to ACTM.

In future works, we will improve the tree aggregation algorithm to detect worms faster with smaller network overhead.

Acknowledgments This work is supported in part by a Grant-in-Aid for the Global Center of Excellence for high-Level Global Cooperation for Leading-Edge Platform on Access Spaces from the Ministry of Education, Culture, Sport, Science and Technology in Japan, a grant from Secom Science and Technology Foundation, and a Grant-in-Aid for Scientific Research (C), 2006, 1850063.

References

1) Aiello, W., et al.: Analysis of communities of interest in data networks, *Proc. Passive and Active Measurement Workshop 2005* (2005).

2) Briesemeister, L.: Microscopic simulation of a group defense strategy, *Proc. 19th PADS*, pp.254–261 (2005).

3) Zou, C.C., et al.: Monitoring and early warning for internet worms, *Proc. 10th ACM CCS* (2003).

4) Ellis, D., et al.: A behavioral approach to worm detection, *Proc. ACM WORM 2004*, pp.43–53 (2004).

5) Gu, G., et al.: Worm detection, early warning and response based on local victim information, *Proc. IEEE 20th ACSAC*, pp.136–145 (2004).

6) Shigeno, H., et al.: A Distributed Worm Detection Method based on ACTM, *IPSJ SIG T.R. 2007-DPS-130, 2007-CSEC-36*, pp.201–

206 (2007).
7) Williamson, M.: Throttling Viruses: Restricting Propagation to Defeat Malicious Mobile Code, *T.R. HPL-2002-172* (2002).
8) Kawaguchi, N., et al.: ACTM: Anomaly Connection Tree Method for Detection of Silent Worms, *IPSJ Journal*, Vol.48, No.2, pp.614–643 (2007).
9) Kawaguchi, N., et al.: d-ACTM: Distributed Anomaly Connection Tree Method to detect Silent Worms, *Proc. 26th IEEE IPCCC* (*Malware'07 Track*) (2007).
10) McDaniel, P., et al.: Enterprise security: A community of interest based approach, *Proc. NDSS 2006* (2006).
11) Pang, R., et al.: A first look at modern enterprise traffic, *Proc. IMC 2006* (2005).
12) Antonatos, S., et al.: Defending against hitlist worms using network address space randomization, *Proc. WORM 2005* (2005).
13) Senthilkumar, G., et al.: A distributed host-based worm detection system, *Proc. ACM SIGCOMM WORKSHOPS* (2006).
14) Staniford, S., et al.: How to own the internet in your spare time, *Proc. 11th USENIX Security Symposium* (2002).
15) Staniford, S., et al.: The top speed of flash worms, *Proc. ACM WORM 2004*, pp.33–42 (2004).
16) Staniford-Chen, S., et al.: Grids: A graph-based intrusion detection system for large networks, *Proc. 19th National Information Systems Security Conference*, pp.361–370 (1996).
17) Toth, T., et al.: Connection-history based anomaly detection, *Proc. 3rd IEEE Information Assurance Workshop* (2002).
18) Sekar, V., et al.: Toward a framework for internet forensic analysis, *Proc. HotNets-III* (2004).
19) Yegeswaran, V., et al.: Global intrusion detection in the domino overlay system, *Proc. NDSS 2004* (2004).
20) Xie, Y., et al.: Worm origin identification using random moonwalks, *Proc. 2005 IEEE S&P*, pp.242–256 (2005).

(Received May 18, 2007)
(Accepted November 6, 2007)
(Online version of this article can be found in the IPSJ Digital Courier, Vol.4, pp.79–90.)

Editor's Recommendation

This paper proposes a distributed detection method of Hit-list worms that use address lists of vulnerable hosts for propagation. In the proposed method, distributed IDSes which are deployed in enterprise networks cooperate to detect Virtual Anomaly Connection Trees based on the outgoing connection models of internal hosts. The approach is novel and useful for its ability to quickly detect hit-list worms that can evade most of existing approaches. The achievement of this paper therefore gives a significant contribution to the further development of the effective worm detection techniques. I recommend this paper for publication in the IPSJ Journal.

Nobutaka Kawaguchi received the B.S. in the Department of Information and Computer Science from Keio University, Japan in 2003, the M.S. degree in Open and Environment Systems from Keio University in 2005. He is currently working toward the Ph.D. degree in Open and Environment Systems at Keio University. His research interests include Intrusion Detection, Digital Forensics and Security Visualization.

Hiroshi Shigeno received the B.S., M.E. and Ph.D. degrees in instrumentation engineering from Keio University, Japan in 1990, 1992 and 1997 respectively. Since 1997, he has been with the Department Information and Computer Science, Keio University, where he is currently an associate professor. His current research interests include computer networking architecture and protocols, mobile and ubiquitous computing, and agent computing and communications.

Ken-ichi Okada received the B.S., M.E. and Ph.D. degrees in instrumentation engineering from Keio University, in 1973, 1975 and 1982 respectively. He is currently a professor in the Department of Information and Computer Science at Keio University. His research interests include CSCW, groupware, human computer interaction and mobile computing. He is a member of IEEE, ACM, and IPSJ. He was a chair of SIGGW, a chief editor of IPSJ Journal, and an editor of IEICE Transactions. Dr. Okada received the IPSJ Best Paper Award in 1995, 2001 and IPSJ 40th Anniversary Paper Award in 2000.