VMITN: A Novel Intrusion Tolerance Architecture for Treating the Rapid Propagation of Malicious Programs

Wen-Chun Sun and Yi-Ming Chen
Department of Information Management, National Central University
nivek, cym@mgt.ncu.edu.tw

ABSTRACT

Today's understaffed IT departments face a daunting security challenge – protect the enterprise from new, unknown threats. The most damaging threats, unfortunately, are coming from fatal malicious programs, such as zero day worms and viruses, which are hard to be stopped by traditional security mechanisms. Therefore, instead of trying to prevent every single intrusion, in this paper, we adopt a novel system architecture which will tolerate new worm attack temperately until administrator removes the vulnerability. With a set of intrusion pattern recognition mechanisms and the virtual machine technology, the proposed VMITN (Virtual Machine based Intrusion Tolerance Network) is able to achieve the goal of intrusion tolerance. We have implemented a prototype of VMITN. We present the design, implementation and evaluation of this prototype system. Our experiments in an emulation network proved the reliability and survivability of VMITN under Code Red worm attack.

1: INTRODUCTIONS

A traditional approach for security engineering is establishing a preventive barrier, like a firewall or an IPS, to protect the infrastructure resources from intruders. Unfortunate, with the increase of network attack incidents, the efficiency of a single barrier unit is not good enough to prevent attacks from sophisticated new attacking skills [4]. New breeds of computer worms [9] such as SQL Slammer, MS Blaster and Slapper worms, infect thousands of computers and cause massive denial of service outages on the Internet.

In contrast to pursue the nearly impossibility of a perfect barrier, many researchers against rapid propagation threats are working on intrusion tolerance in recently years [16]. An intrusion tolerant system is one that can avoid system failure, continue to function correctly, and provide the intended services to users in a timely manner even under attack. In other words, intrusion tolerance focuses on establishing system dependability, which is defined as "a property of a computer system such that reliance can justifiably be placed on the service it delivers" [1]. In the well-known fault-error-failure sequence as show in Fig 1 [23], the failure is defined as a delivered service deviates from fulfilling the system function and fault tolerance is one important factor to compensate the disaster derived from system failure. Intrusion tolerance inherits the same scheme to provide the system dependability under network attacks.

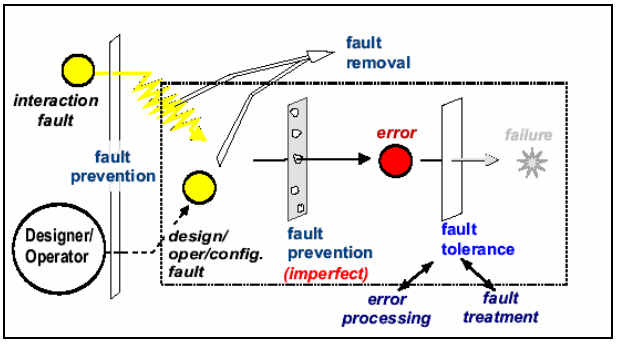

Fig 1: Achieving dependability [23]

In this paper we present a novel architecture named VMITN (Virtual Machine based Intrusion Tolerance Network) to treat the rapid propagation of malicious programs. VMITN applied the technologies of OOB (Out-of-Band) network and virtual machine. An OOB network is based on a set of physically independent channels which connect to the second NIC (Network Interface Cared) of each network node to guarantee the control of network even the primary network, which connects to the primary NIC of each node, is under attacks. The reason why the OOB network could survive the attack for primary network is due to the use of virtual machine that uses the VMM (Virtual Machine Monitor) [10] to isolate the native OS and guest OS which are connected to the OOB network and primary network respectively.

One more important factor to the success of intrusion tolerance is the capability of resisting the mass destruction even though the malicious worms or virus propagates very fast. Therefore, we developed a new algorithm for VMITN to quickly recognize attack patterns and then seamlessly refresh infected hosts. Fig 2 shows a typical worm chart in which the bold line presents disaster caused by a worm in the life cycle [3]. In this figure, the number of infected hosts grows fast and keeps high until the stage of manual blocked takes place. However, with the redundancy of OOB network

and the quickly switch of guest OS, the VMITN could withhold the malicious worm propagations (as the dot line illustrated in Fig 2) to ease the work of manual intervention.

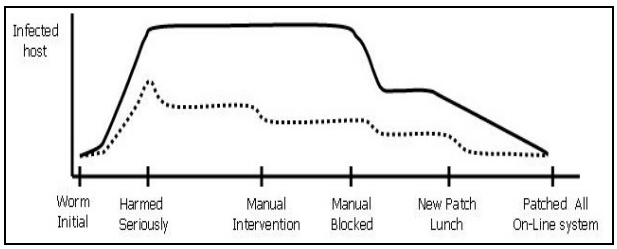

Fig 2 : Worm Propagation Chart [3]

In this paper is divided into 5 sections. Section 2 states relevant approaches in intrusion tolerance technology and projects applied VM technologies. The prototype systems and the design of countermeasures are discussed in Section 3. In Section 4, some evaluations are performed in our experimental network. Section 5 gives our conclusions and explores future research directions.

2: RELATED WORK

The term "intrusion tolerance" appears originally in a paper by Fraga and Powell [8]. A number of isolated intrusion tolerance protocols and systems were emerged afterward. For example, the scheme –Fragmentation-Redundancy-Scattering– was used by people who develop an intrusion-tolerant distributed server composed by a set of insecure sites [5]. In the meantime, the Byzantine quorum systems were used to build security servers and data storage [17]. A threshold crypto technology was proposed for intrusion-tolerant file storage and the distribution of user secrets [18]. Recently, two famous intrusion tolerance projects are OASIS [14] and MAFTIA [19].

Though previous researches in intrusion tolerance obtained many promising results [14] [19]. Most of them suffer a shortcoming that their designs are based on a strong assumption that potential vulnerable components are predictable [24]. But the central problem in facing new malicious programs is that the attacking targets and tricks are usually contrary to expectation [12]. To solve this problem, some researches proposed the concept of network topology dependability [11] which could resist the mass attacks from malicious programs. The VMITN extends this concept and adopts the VM technology to tolerate system level intrusion.

VMM provides a very good trusted computing base as it has narrow interfaces and small size to isolate native OS and guest OS [6]. In case of a guest OS becoming fatal, there is no impact to the native OS. Lots of researches, such as Collapsar [10] and Revirt [6], utilize VM technology in network security due to the following features of VM: (1) restricting hardware resource access, (2) monitoring all activities on a guest OS, and (3) forcing a guest OS to power off or changing peripheral devices. In other words, the VM provides high controllability and monitorability to native OS to reconfigure and recover guest OS from failover.

3: DESIGN of VMITN

In this section, we illustrate the VMITN architecture, describe the implementation of its key components, and introduce two mechanisms to practice its performance. The detail implementation of VMITN can be referred in the paper [2].

3.1: Architecture

The full system architecture of VMITN is shown in Fig 3. The architecture consists of two major components: the FES (Front-End System) with VM enabled is running on each network node and the SMC (Security Management Center) is a centralized operation platform for network administrator to control and monitor all FESs. The OOB network connects the FES and SMC.

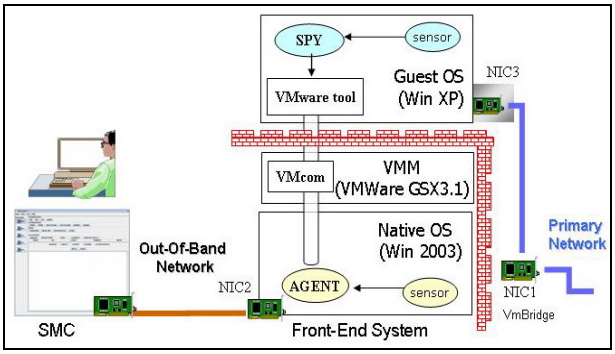

Fig 3: System Architecture

3.2: Front End System

FES is the key element in the VMITN design. As shown in Fig 3, we use the commercial VMWare GSX 3.1 as the VMM [22]. NIC3 is a virtual equipment on a guest OS and serves on a primary network while the physical NIC1 is simulated as a bridge to prevent hackers to identify, detect or invade native OS. The physical NIC2 connects to the OOB network.

As shown in Fig 4, there are two programs implement the function of FES listed below.

SPY: Residing in the guest OS, it not only contains a HIDS (Host-based Intrusion Detection) to observe and report health condition but also has the capability to reconfigure the guest OS.

AGENT: Residing in the native OS, it forwards messages between SMC and SPY and controls a guest OS while necessary.

The isolation between the guest OS and the native OS is guaranteed by the VmCom which is an external control interface of a guest OS and becomes the only

channel between SPY and AGENT. The VM Switcher within the AGENT program is used to quickly replace a compromised guest OS by a fresh new one and will be described below.

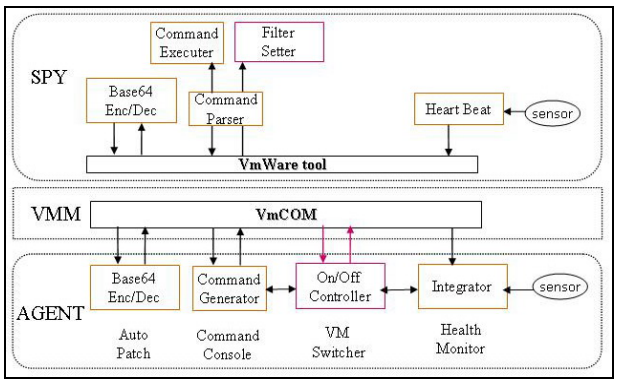

Fig 4: Component in Front-End System

3.3: Rapidly Hand Over

This is an important function of intrusion tolerance. Once a guest OS is infected by a malicious program, the VMITN has to replace the infected OS by a new one immediately to shorten the service down time and prevent the infected one becoming a zombie to attack other suspicious hosts on the Internet. Therefore, one guest OS replication is always ready on line as shown in Fig 5. A VM Switcher module residing in the AGENT can hide a backup guest OS in peacetime and support seamless hand over as soon as a system fault is detected.

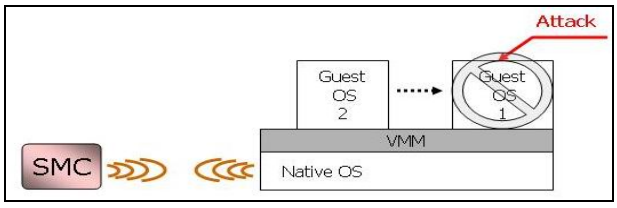

Fig 5: VM Hand Over

3.4: Quickly Attack Pattern Learning

To counter the fast propagation of malicious programs, the early detection of attack traffic is very important [3][13]. For VMITN, which has a SMC to centrally analyze the reports from AGENTs through OOB network, a simple algorithm is developed to block the attack packets in the very beginning. To make the algorithm effect, we make two assumptions: (1) Attack packets from single malicious program must attack one particular vulnerability on specific port of a victim host. Therefore, even mutation program dynamically alternates packet content, some specific pattern still exist in every attack packet. (2) Malicious programs must generate the overwhelming volume packets than other legitimate traffic. For example, a SQL slammer worm [7] sends a great number of 400 bytes-UDP packets to scan port 1434.

Our algorithm consists of two stages: the first stage selects the possible attack packets by the LRU (Least Recently Used) algorithm which ignores large part of normal packets by the use of aging and some threshold value λ. The second stage continues to select the most specific attack pattern matched with the new possible attack packet filtered out in the first stage. After this stage the frequency number of the most matched pattern will be increase by 1 and if the frequency number is larger than some threshold value θ, the most matched pattern will be sent out by SMC to all AGENTs resided in each host to block the attack packets. Fig 6 shows the details of the algorithm.

```
Main ()
{
   For each new packet arrived, extract the packet head
   Separate the src_ip, dst_ip and port to three arrays and
       update the count if it matches to some previous
       records.
   If (updated count > λ ) {
          call Attack_Pattern_Matching_Function( )
          // find the most specific pattern corresponding to
          // the packet and update the Freq of the former.
   }
   If ( Freq > θ ) {
          ask the AGENT to block the attack packet with
          the pattern found.
   }
   exit(0);
}


Attack_Pattern_Matching_Function ()
{
   i=0;
   read first record in table
   while( ! end of table) {
          compare new packet with current record;
          M [i]=# of attrib equivalence;
          i++;
          next record;
    }

    j= index of record that maximum value M[];
    // find the most specific pattern in current table

    read record[j] ;
    if ( seed==1) {
          freq++;
    } else {
          for each attrib with different value from new
               record, set the attrib='*';
          set seed=1;
          freq=1;
    }
}
```

Fig 6: Attack Pattern Learning Algorithm

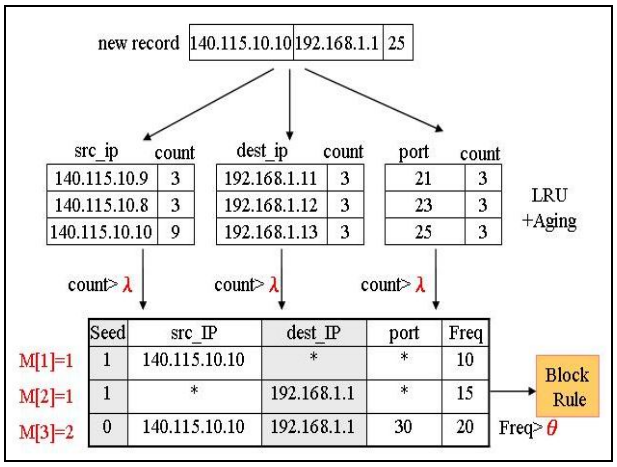

Fig 7: An Example of Attack Pattern Match Algorithm

An example of our algorithm is illustrated in Fig 7. In this figure, when the packet with head of {src_ip, dst_ip, port} = {140.115.10.10, 192.168.1.1, 25} arrives, it is fed into the LRU + Aging module and it makes the last record of the src_ip array increase its count to be larger than 10 (λ). In the consequence, the system compares the packet with the pattern existed in the rough set table. As it most matches to the third record, it increases the Freq number by 1.

4: PERFORMANCE EVALUATION

We have implemented a set of FESs as well as a SMC to establish an experimental network and then we did a number of experiments to measure the performance.

4.1: Experimental Network

As shown in Fig 8, the experimental network is separated into 4 sub networks by 6 routers. R1~R4 represent edge routers, while R5 and R6 represent core routers. There are 7 physical hosts deployed in the network. One of those hosts is designated as SMC with one wireless NIC; the others install VM-based FES with 2 network interfaces (one of which is wireless NIC). Default wired NIC connects to the primary network, while the secondary 802.11 wireless NIC connects to the SMC through OOB network. We installed Windows 2003 server as native OS, and Windows XP as the guest OS. Administrators operate on the SMC to control the whole network and the experiment processes. All of the routers and hosts use class B private IP addresses.

4.2: System Performance

To measure how long a SMC needed to restart a guest OS while the latter is under attack, We list the average spending time in each detail operation within the FES in Table 1. It is noted that the time durations of "SMC Stop Guest OS" and "Restart Guest OS" depend on different OS environments. Administrators must do some manual jobs to fix guest OSs. Doing so might take variable times in different attack cases. Therefore, they are not counted in this table. The rapid handover technology described in Section 3.3 can exclude the non-predicable time and thus can shorten the service interrupt time within about 2 seconds.

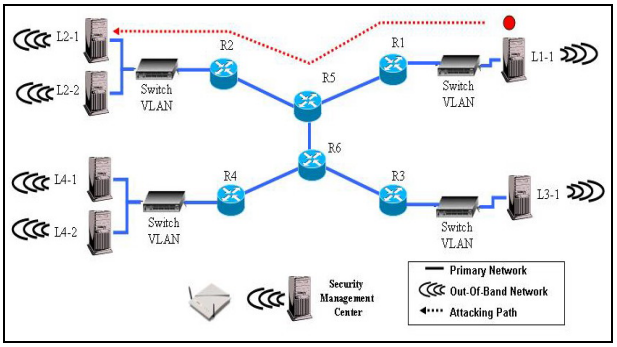

Fig 8: Experimental Network

Table 1: Service Interrupt Time

Item	Time (ms)
Sensor alarm intrusion happened	91ms
SPY reported to AGENT	867ms
AGENT forward to SMC	300ms
Display alarm in SMC GUI	49ms
SMC Shutdown Guest OS	depends on case
Mount virtual disk	471ms
Manual job (scan and clean)	depends on case
Un-mount disk	268 ms
SMC restart Guest OS	depends on case
Total	2,046 ms

4.3: Sensitivity of θ

In the "Attack Pattern Learning" algorithm mentioned before, the sensitive of θ is a very important factor to detect the outbreak of attacking packets. The administrator is responsible for adjusting θ. We use the TFN2K [20] program to emulate a DDOS attack from host L1-1 to other hosts and measure the learning time in different attack traffic loads. In Fig 9, the threshold θ is set to three different levels: 100, 200 and 300. Unsurprisingly, the high threshold value whish represents the FESs is less sensitive to the existing attack of packets requires longer detection time. On the other hand, high sensitive detection can block malicious program under 3 seconds even in low attack packet rate (eg: < 250 packet/sec).

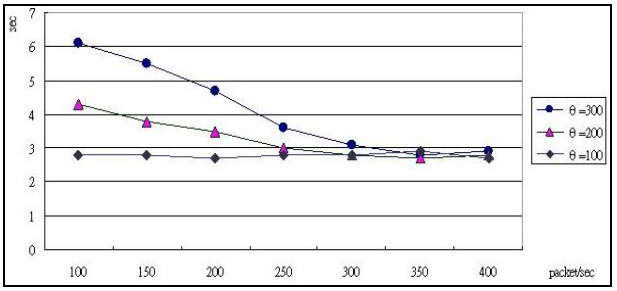

Fig 9 : Learning Algorithm Sensitivity

4.4: Intrusion Torrance to Rapid Propagation Threat

To measure the intrusion tolerance capability of VMITN in facing a rapid propagation malicious program, we develop a program to emulate the behavior of Code Red worm The program sends 1826 packet/sec to port 80 with localized scanning strategy [25], that choices a random address within class B in probability 3/8, within class A in probability 1/2, others for the whole Internet. We lunched the Code Red worm emulator at one of hosts of the experimental network in initial and observed there were distinct propagation behaviors in different countermeasures.

In Fig 10, the bold line represents the average propagation speed in 20 experiments without any countermeasure enabled. It is noted that all six hosts are infected within 6 seconds. The other line represents the worst case within 20 experiments when "Rapidly Hand Over" countermeasure is enabled. An infected host propagates the worm before it is refreshed and it might be re-infected after refreshed because the vulnerability still exists. If the speed of refresh is faster than the propagation speed of worm, the worm would be finally cleaned from the network. In the 20 experiments we performed, infected hosts are controlled under 4 or less hosts and worm is cleaned in average 7.8 seconds.

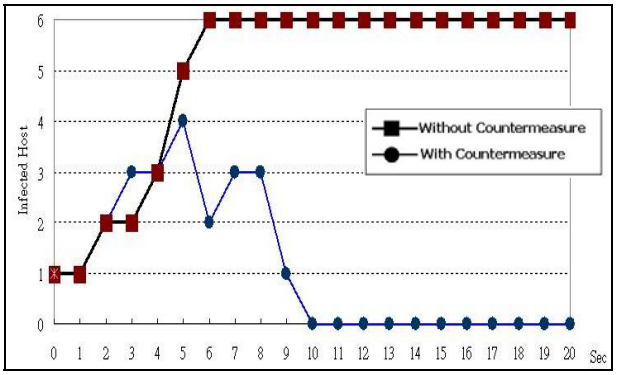

Fig 10 : Rapidly Hand Over Resists Code Red

In Fig 11, the dotted line represents the worst case within 20 experiments when "Attack Pattern Learning" technology is enabled to block continuously infecting next suspicious victims. Due to the lack of the clean mechanism, worm propagation is faster then Fig 10 at beginning, but propagation stopped once the scan packets pattern is learned. The experiments showed that the algorithm can detect the disproportional scanning packets appearing in the network and correctly figure out the pattern in average 3.7 seconds. This in turn allowed the worm attacks 4 hosts at most.

In Fig 12, both “Rapid Hand Over” and “Attack Pattern Learning” are enabled at the same time. Due to "Rapidly Hand Over" is started during the pattern learning period, the number of attacking packet is decreased than experiment in Fig 11. For this reason, the learning efficiency is slowed down to average 5.7 seconds in 20 experiments. The worm is cleaned in average 6.1 seconds; however, the maximum number of affected hosts in these experiments is reduced to 3 hosts.

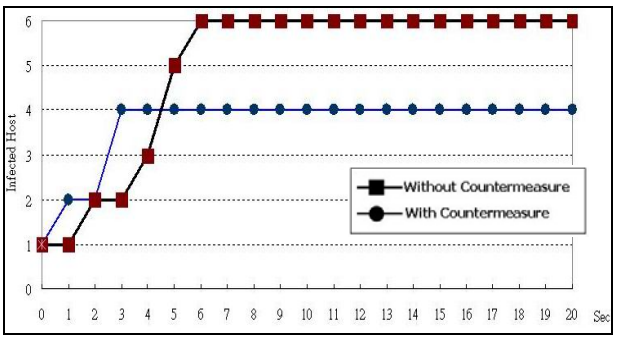

Fig 11 : Attack Pattern Learning Resist Code Red

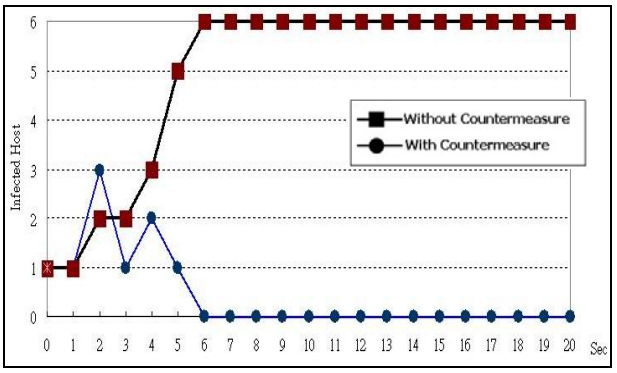

Fig 12 : Both Countermeasures Enabled

We measured the availability of VMITN by accumulate average health hosts with correct service in 20 seconds experiment when both “Rapid Hand Over” and “Attack Pattern Learning” are enabled. The result showed average 86% hosts are health to service after Code Red worm lunched. Even in the peak of worm infection, only 50% victims are infected.

5: CONCLUSIONS

Impeding new attacks is a key challenge in current network security society. Intrusion tolerance is a new approach, which tolerates the existence of vulnerabilities while keeps the mission critical applications running.

In this paper, we present the design and implementation of a novel intrusion tolerant architecture which applies VM-based and OOB network to support reliable control even though the primary network is under severe attack. Our experiments in an isolated network show that by “Rapid Hand Over” and “Attack Pattern Learning” technologies, the VMITN could keep up to 86% service capability while limit the victim number under 50% of total hosts in facing a new malicious worm.

It is worth to continue some points in this research in the future: (1) FESs are expected to be migrated from commercial software VMWare to open-source VM platforms, like UML [21]. (2) In our implementation,

the SMC can not handle GUI events in the guest OS. The handling of GUI event should be implemented in the future. (3) Currently we only emphasize on the handover of OS, the performance decrease of application due to the handover of OS should be concerned and improved.

6: REFERENCES

[1] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, “Basic concepts and taxonomy of dependable and secure computing”, IEEE Trans. Depend. Secure Comput., vol. 1, no. 1, pp. 11–33, Jan.–Mar. 2004.

[2] Yi-Ming Chen, Wen-Chen Sun, "A reliable remote control platform to oppose a large scale attacking", Symposium on Applications of Information, and Communication Technology (SAICT), Kaohsiung, Taiwan, June 2006.

[3] Xuan Chen and John Heidemann, "Detecting Early Worm Propagation through Packet Matching", USC Information Sciences Institute Technical Report, ISI-TR-2004-585, February 2004.

[4] Yves Deswarte, David Powell, "Internet Security: An Intrusion-Tolerance Approach", Proceedings of the IEEE, VOL. 94, NO. 2, 2006

[5] Y. Deswarte, L. Blain, J.C. Fabre, "Intrusion tolerance in distributed computing systems", In: Proceedings of the IEEE Symposium on Research in Security and Privacy.

[6] G. Dunlap, S. King, S. Cinar, M. Basrai, P. Chen, "ReVirt: Enabling Intrusion Analysis through Virtual-Machine Logging and Replay", Proceedings of the 2002 Symposium on OSDI.

[7] eEye Digital Security, "SQL Worm Analysis", http://www.eeye.comhtml/Research/Advisories/AL20020522.html

[8] J.S. Fraga, D. Powell, "A fault- and intrusion-tolerant file system" In: Proceedings of the 3rd International Conference on Computer Security. 203–218, 1985.

[9] Eul Gyu Im, Jung Taek Seo, Dong Soo Kim, Yong Ho Song, Yong Su Park. "Hybrid Modeling for Large-Scale Worm Propagation Simulations." In Proceedings of the 2006 IEEE ISI 2006, San Diego, CA, USA. May 2006

[10] Xuxian Jiang, Dongyan Xu "Collapsar: A VM-Based Architecture for Network Attack Detention Center", Proceedings of 13th USENIX Security Symposium (Security'04), San Diego, CA, August, 2004

[11] Havard Johansen, Andre Allavena, Robbert van Renesse, "Fireflies: Scalable support for intrusiontolerant network overlays", In Proceedings of Eurosys 2006.

[12] J. E. Just, J. C. Reynolds, L. A. Clough, M. Danforth3, K. N. Levitt, R. Maglich, J. Rowe, "Learning Unknown Attacks—A Start,” Proceedings of the 5th International Symposium, 2002.

[13] Yoohwan Kim, Wing Cheong Lau, Mooi Choo Chuah, H. Jonathan Chao, "PacketScore: A Statistics-Based Packet Filtering Scheme against Distributed Denial-of-Service Attacks," IEEE Transactions on Dependable and Secure Computing, vol. 3, no. 2, pp. 141-155, 2006.

[14] J. H. Lala, "OASIS—Foundations of Intrusion-Tolerant Systems", Los Alamitos, IEEE Comput. Sci., 2004.

[15] Fu-Yuan Lee, Shiuhpyng Shieh, "Scalable and lightweight key distribution for secure group communications", International Journal of Network Management, Volume 14 Issue 3, May 2004.

[16] Mirage Network, "Combating Rapidly Propagating Threats From the Internal Network", 2003 http://www.appliednetsec.com/productresources/mirage/Combating RPTs from the Internal Network 10pages.pdf

[17] D. Malkhi, M.K. Reiter, D. Tulone, E. Ziskind, "Persistent objects in the Fleet system" In: Proceedings of the 2nd DARPA Information Survivability Conference and Exposition (DISCEX II). 2001

[18] F. B. Schneider and M. A. Marsh, “CODEX: A robust and secure secret distribution system,” IEEE Trans. Depend. Secure Computer, vol. 1, no. 1, 2004.

[19] R. Stroud, I. Welch, J. Warne, and P. Ryan, "A qualititative analysis of the intrusion-tolerant capabilities of the MAFTIA architecture", Int. Conf. Dependable Systems and Networks (DSN), Florence, Italy, 2004.

[20] Tribal Flood Network2K http://staff.washington.edu/dittrich/misc/tfn.analysis

[21] User Mode Linux. http://user-mode-linux.sourceforge.net

[22] VMWare http://www.VMWare.com/

[23] P. Verssimo, N. F. Neves, and M. P. Correia. "Intrusion-tolerant architectures: Concepts and design". Architecting Dependable Systems, volume 2677 of Lecture Notes in Computer Science. Springer-Verlag, 2003.

[24] F. Wang, R. Uppalli, "SITAR: A Scalable Intrusion-Tolerant Architecture for Distributed Services", In Volume II of the Proceedings of DISCEX III, pages 153--155, April 2003.

[25] C.C. Zou, W. Gong, D. Towsley, "Code Red worm propagation modeling and analysis", In: Proc. of the 9th ACM Symp. on Computer and Communication Security. Washington, 2002. 138~147.