Active versus Passive Malware Collection

Ying-Dar Lin, Chia-Yin Lee, Yu-Sung Wu, Pei-Hsiu Ho, and Fu-Yu Wang, *National Chiao Tung University*
Yi-Lang Tsai, *National Center for High-Performance Computing*

An exploration of active and passive malware honeypots reveals that the two systems yield vastly different malware collections and that peer-to-peer file sharing is an important, but often overlooked, malware source.

Malware is second only to network-intrusion techniques as a threat to Internet security.1,2 Viruses, worms, Trojans, back doors, rootkits, and, more recently, bots3 can spread indirectly through users who download files from a malicious URL or share files in a peer-to-peer (P2P) network, or directly by exploiting vulnerabilities and entering the host.

Malware countermeasures typically take the form of automated collection through passive or active honeypots. *Passive* honeypots[4,5] gather samples by luring the malware to a specific system. *Active* honeypots collect samples by deliberately linking to or opening malicious URLs or files. To date, little information is available on the differences between the collected malware, which might answer questions such as, are the malware samples sufficiently disjoint to warrant using both active and passive systems?

Developing automated collection software requires understanding exactly how malware behaves in the host and network, how it propagates, and how it might evade techniques to detect and collect it. To better understand malware sources and behavior, we developed Honey-Inspector, an active honeypot system that we made open source (http://honeyinspector.sourceforge.net/).

Honey-Inspector collects malware from malicious websites as well as from shared peer-to-peer (P2P) files, which are becoming a popular channel for malware propagation. Collecting malware from P2P file sharing generates volumes of suspicious programs, which Honey-Inspector prunes uses antivirus scanners. The remaining programs execute in a virtual machine, and Honey-Inspector analyzes how they alter the host or compromise the network. Through the virtual machine, a closed execution environment, we could extensively examine the detected malware's characteristics.

We also conducted tests to compare malware collection samples from Honey-Inspector and from the passive honeypot system at the National Center for High-Performance Computing (NCHC) in Taiwan. The system, which operates as part of the international honeynet project (www.honeynet.org/node/157), consists of 3,600 server honeypots

Table 1. Comparison of client honeypots and Honey-Inspector.

Item	HoneyMonkey	HoneyClient	Capture-HPC	Honey-Inspector
Collection sources				
Malicious URLs	✓	✓	✓	✓
Email	✓			
Shared P2P files				✓
Observation of malware's host behavior				
File system	✓	✓	✓	✓
Registry	✓	✓	✓	✓
System configuration	✓			✓
System process		✓	✓	
Observation of malware's network behavior				
Network traffic	✓	✓	✓	✓

deployed across nine academic networks. Network traffic volume is approximately 100 Gbytes per hour.

We conducted our experiment to answer three questions:

- What types, or distribution, of malware do the two approaches collect, and are the resulting collections disjoint or overlapping?
- How timely is malware capture? For example, can either approach capture malware that does not yet have a defined signature in a scanner database?
- How strong is the degree of host and network activity (activeness) of the malware that each system captures?

Knowing the answers to these questions will enable us to more intelligently plan strategies that can collect more and newer malware and detect it earlier.

MALWARE COLLECTION

Honeypots can be on either the server or client side. A *server* honeypot passively lures a malware attack by exhibiting a variety of popular services with set vulnerabilities.[4,5] The passive honeypot system at NCHC, for example, uses vulnerabilities from the US National Institute of Standards and Technology's National Vulnerability Database (http://nvd.nist.gov/). The main disadvantage is that passive honeypots often cannot capture malware designed to exploit client-side vulnerabilities.

Client honeypots evolved in 2002 primarily to address this gap. A *client* honeypot typically comprises deliberately vulnerable client software that actively interacts with Internet services to attract malware designed to attack the client side.

Client honeypots are either high or low interaction. High-interaction honeypots come with a full client software stack. The vulnerabilities they exhibit are more realistic, but the honeypot setup is more complex, and runtime overhead is high. Examples of high interaction honeypots are Honey-Client (www.honeyclient.org/trac/), HoneyMonkey,[7] and Capture-HPC (https://projects.honeynet.org/capture-hpc/). Honey-Inspector is also a high-interaction honeypot. Its virtual machine imitates application behaviors and responses realistically enough to deceive the malware into seeing it as an actual system.

Low-interaction honeypots[12] emulate client software vulnerabilities instead of running a client software stack. They are easier to set up but cannot always capture malware that targets vulnerabilities outside the emulation.

Table 1 gives the comparative characteristics of Honey-Client, Honey Monkey, Capture HPC, and Honey-Inspector. Only HoneyMonkey and Honey-Inspector collect malware from sources other than malicious URLs. Honey-Inspector adds P2P but omits email collection primarily because malware spreading via email is passive in nature in the sense that a victim does not actively solicit mails carrying malicious contents.

MALWARE DETECTION AND ANALYSIS

Malware detection is based on the malware's behavior or on its signature.[13] *Behavior-based* detection can detect malware that is previously unknown and thus has no recognizable signature. The challenge in this detection approach is determining at runtime what features to observe, which slows detection speed.

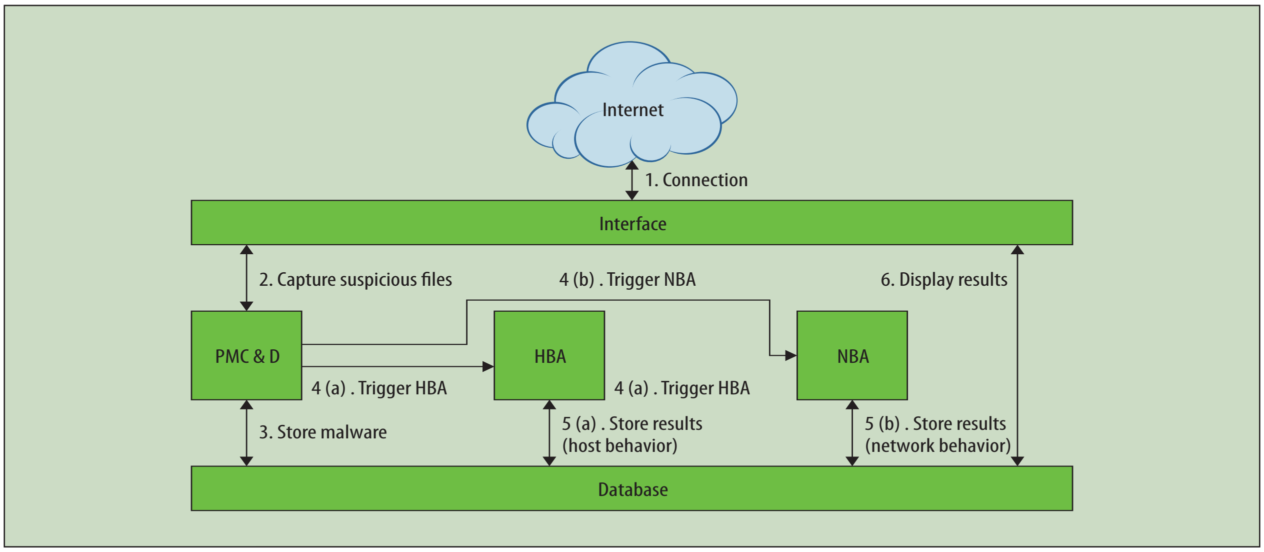

Figure 1. Malware collection, detection, and analysis in Honey-Inspector. After (1) connecting to the Internet, Honey-Inspector begins malware capture through the Proactive Malware Capture and Detection (PMC&D) module, which (2) collects suspicious files and (3) stores them in the database. The module then (4) triggers the Host Behavior Analysis (HBA) and Network Behavior Analysis (NBA) modules to analyze host and network behavior, respectively, and (5) store results in the database. (6) The interface displays results.

Signature-based detection tries to characterize malware as a signature, which it stores in a signature database. By reviewing the database, it can quickly detect if a program is malicious (has one of the established signatures), but it cannot detect unknown malware whose signature is not defined in the database.

Client honeypot systems tend to use behavior-based detection only because malware known by signature-based detection systems would present little threat to most clients due to the widespread use of anti-virus software on the clients. Honey-Inspector also uses behavior-based detection but inserts signature-based scanning before detection. Signature-based scanners can reduce the volumes of suspicious programs that amass from adding P2P file sharing as a collection source, leaving less for the slower behavior-based detection to sift through.

INSIDE HONEY-INSPECTOR

As Figure 1 shows, three modules handle the workflow in Honey-Inspector: Proactive Malware Capture and Detection (PMC&D), Host Behavior Analysis (HBA), and Network Behavior Analysis (NBA). The PMC&D module proactively collects suspicious samples, such as executable files, from P2P file sharing and Web browsing. It then uses scanners to divide the samples into benign or malicious, storing malicious samples in the database.

The HBA and NBA modules observe how malware behaves after it executes on the virtual machine. Does it modify the file system or registry on the virtual machine? Does it launch network attacks? The HBA module compares snapshots of the virtual machine's file system and registry before and after execution. Meanwhile, the NBA module sniffs and records network traffic in the virtual machine environment. If the HBA and NBA modules do not reveal any malicious behavior, Honey-Inspector considers the program benign.

Honey-Inspector does not compromise the Internet in any way. The PMC&D module does not invoke sample execution, and the browsing rate is controlled. Although the NBA and HBA modules require executing a collected sample, our analysis environment—our group of virtual machines—is a closed network.

Proactive Malware Capture and Detection

Figure 2a shows the activities in the PMC&D module. Currently, we use the Avast, Avira AntiVir, Kaspersky, and Nod32 antivirus scanners because their false negative rates are less than 2.4 percent.[14] If any of these scanners suspects the program is malware, the PMC&D module stores the sample and detection results in the database.

Host Behavior Analysis

As Figure 2b, shows, the HBA module sets up a new virtual machine and copies the clean registry and file system. It then executes the suspicious sample and uses the DiffReg and DiffFS modules to check for infection in the registry and file system and identify any modifications. By comparing these modifications against a clean virtual machine image, we can identify the malware. As a final task, the HBA module stores results (before and after execution) in the database.

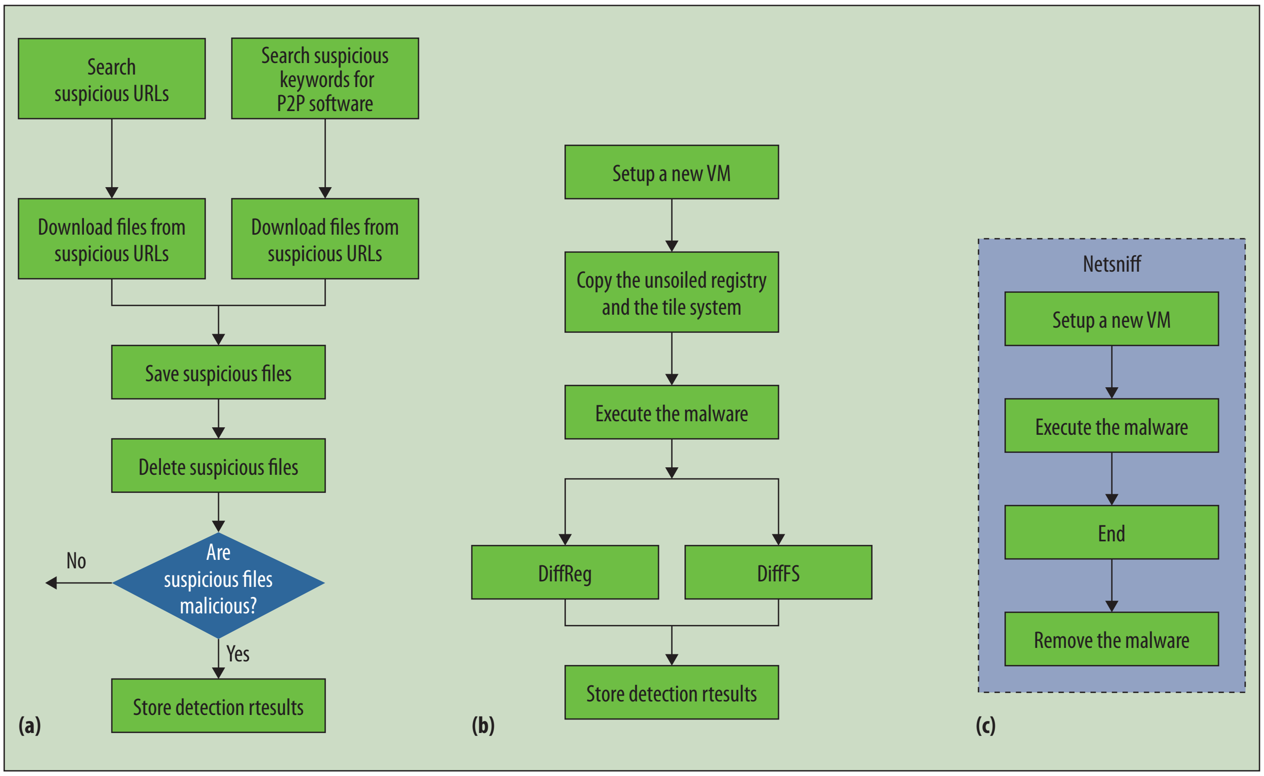

Figure 2. Workflow in Honey-Inspector's three modules. The (a) PMC&D module's main tasks are to collect malware samples, send them to antivirus scanners, and store suspected malware in the database. The (b) HBA and (c) NBA modules analyze the suspected malware's behavior in the host and network after it executes in the virtual machine.

Network Behavior Analysis

Figure 2c depicts the tasks to detect if a malicious program has generated network traffic. The NBA module sets up a new virtual machine and executes a suspicious program. It then uses the Netsniff module to monitor the virtual machine's network traffic. If a suspicious program generates network traffic, such as sending an email or ICMP packet, Netsniff will sniff the packet traces and store them in the database.

EXPERIMENTAL RESULTS

To gather data on the malware from Honey-Inspector and the passive honeypot system at NCHC, we ran Honey-Inspector on a PC with several virtual machines and observed malware collection, detection, and analysis for a month. During the same month, we collected 354 unique malware samples from the passive honeypot system. We observed 800 unique malware samples from Honey-Inspector.

Types of captured malware

Figure 3 shows the types of malware the two systems captured. Overall, Honey-Inspector captured more Trojans but fewer bots than the NCHC system.

Bots made up 79 percent of the passive honeypot system's captured malware, with the remainder comprising worms, Trojans, and other malware types. Bots spread by scanning computer vulnerabilities through the network, making it easy for the passive honeypot system to capture them. In contrast, Honey-Inspector captured mostly Trojans (59 percent), with bots, worms, and other malware making up the remaining 41 percent. Because Honey-Inspector actively seeks potentially malicious binary files from multiple sources, it is more likely to find a Trojan, which hides inside an outwardly harmless program. The passive honeypot system, on the other hand, waits until a Trojan initiates a remote attack, which explains its small percentage of captured Trojans. The passive honeypot system's malware distribution is consistent with the results from a related study.[18]

An analysis of the malware sources revealed that malware from P2P file-sharing software exceeded malware from websites by several orders of magnitude primarily because the method that peers employ to search for content makes them vulnerable. Indeed, P2P file-sharing was the dominant source of malware.

Timeliness

Figure 4 shows the capture time of malware using Honey-Inspector and the passive honeypot system. Time begins when the malware's signature is defined in the antivirus

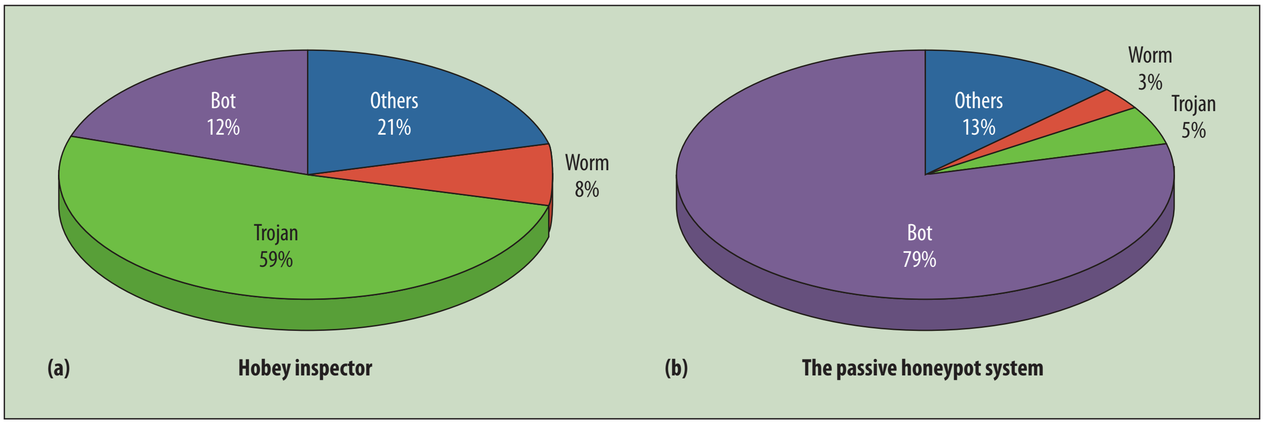

Figure 3. Distribution of captured malware for (a) Honey-Inspector and (b) the passive honeypot system. Honey-Inspector captured more Trojans because it actively seeks potentially malicious binary files, while the passive system captured more bots because bots scan the network for vulnerabilities, such as those in a passive honeypot.

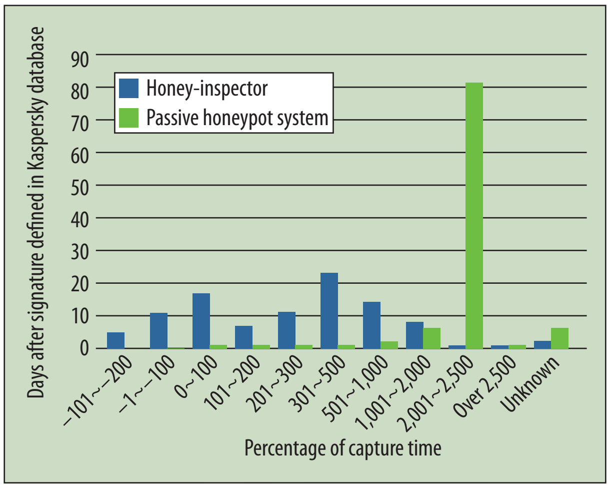

Figure 4. Capture time of malware for Honey-Inspector and NCHC's passive honeypot system. Day 0 represents the day that the signature is defined in the Kaspersky antivirus scanner's database of malware signatures. Unlike the passive honeypot system, Honey-Inspector identified 16 percent of its collected malware before day 0 and often identified malware hundreds of days before then (negative numbers).

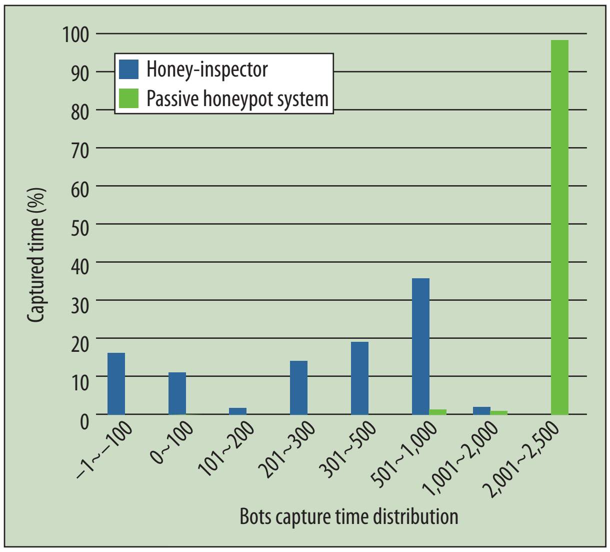

Figure 5. Capture time of bots for Honey-Inspector and the passive honeypot system. Honey-Inspector consistently captured bots earlier.

scanner's database and ends when the collection module first finds the malware. A high number reflects poorly on the collection system because antivirus scanners have known of the malware since day zero. A negative number reflects favorably because the system has found the malware before day zero.

We used Kaspersky's malware signature database in our experiment because Kaspersky is one of the most popular antivirus scanners on the market and (more important), to our knowledge, the only one that publicizes signature definition time for each malware.

Because the antivirus scanner's signature database is updated periodically, we can establish if the antivirus scanner has detected the signature and, if not, how long it takes until a new signature is added. We analyzed this timeline to identify when each scanner first recognized a day-zero malware.

As Figure 4 shows, Honey-Inspector collected 84 percent of its malware after the signature definition appeared in the Kaspersky database. However, it collected 16 percent of its malware before signature definition, as opposed to zero percent for the passive system. These results show that Honey-Inspector is timelier than the passive honeypot system in collecting new malware types.

Figure 5 shows the capture time for collecting bots. Again, Honey-Inspector collected about 84 percent of bots whose signature was already in the Kaspersky database, and 16 percent before the signature appeared. The pas-

Table 2. Percentage of collected malware across four behavior classes

Class	Passive honeypot system	Honey-Inspector
Class A: Malware exhibits both host behavior and network behavior (strong activeness)	67%	48%
Class B: Malware exhibits only network behavior (strong activeness)	31%	29%
Class C: Malware exhibits only host behavior (weak activeness)	1%	12%
Class D: Malware exhibits no behavior (weak activeness)	1%	11%

sive honeypot system collected bots more than 100 days after their signatures were defined in the Kaspersky database, with most bots collected from 2,001 to 2,500 days afterward. Relative to the passive honeypot system, Honey-Inspector can capture much newer bots.

Even more significant is the lack of collection overlap. Fewer than 1 percent of the bots collected were common to both systems. This result could be because a one-month collection time is too short to identify overlaps.

Behavior analysis results

Table 2 shows the results of analyzing the collected malware's behavior, which we grouped into four classes. Each class also reflects the degree of activeness. Malware with network behavior exhibits strong activeness because it can attack remote hosts through the network, while malware with host behavior has a limited ability to do harm.

For the passive system, 98 percent of the samples fell into class A or B with only 2 percent in the remaining classes. One explanation is that the passive system lures malware to infect hosts through network behaviors, so malware is likely to exhibit these behaviors. In comparison, 77 percent of Honey-Inspector's malware samples fell into class A or class B, but 23 percent fell into class C or D. Overall, we can see that Honey-Inspector is more balanced between collecting malware with strong activeness and collecting malware with weak activeness. On the other hand, the passive honeypot system is mostly ineffective in collecting malware with weak activeness.

Comparing Honey-Inspector with the passive server honeypot gave us some answers to the three questions we posed at the beginning of our experiment. We were able to identify the type and distribution of the malware collected by an active (Honey-Inspector) and passive honeypot system, and we learned that the collections are mostly disjoint, although a longer evaluation period might yield different results.

We answered the collection timeliness question as well, finding that Honey-Inspector can capture much newer malware than the passive honeypot system. Finally, we observed differences in the activeness between the collected samples. Honey-Inspector's captured malware exhibited both strong and weak activeness, but the malware from the passive system exhibited only strong activeness. In a nutshell, Honey-Inspector was able to yield a more diverse collection of malware in a shorter period of time. We can also see that the passive honeypot system alone was far from being a comprehensive solution for malware collection.

We plan to extend the malware collection period to discover if we can generalize these findings. We also plan to collect malware from more sources, such as shared links on social networks, and refine malware behavior analysis to enhance the system's overall collection ability. Both Honey-Inspector and the passive honeypot system captured malware with strong activeness. It is unclear whether Honey-Inspector would subsume the passive honeypot system to some degree in this regard. A more detailed study on the composition of the collected malware may help answer the question. C

Acknowledgments

This work was supported in part by the National Communications Commission (NCC), National Science Council (NSC), and Chunghwa Telecom Co. of Taiwan.

References

1. N. Provos et al., "The Ghost in the Browser: Analysis of Web-Based Malware," *Proc. Workshop on Hot Topics in Understanding Botnets* (HotBots 07), 2007; http://static.usenix.org/event/hotbots07/tech/full_papers/provos/provos.pdf.
2. M. van Gundy and H. Chen, "Noncespaces: Using Randomization to Enforce Information Flow Tracking and Thwart Cross-Site Scripting Attacks," *Proc. 16th Ann. Network and Distributed System Security Symp.* (NDSS 09), 2009; https://www.isoc.org/isoc/conferences/ndss/09/pdf/03.pdf.
3. C. Kanich et al., "The Heisenbot Uncertainty Problem: Challenges in Separating Bots from Chaff," Proc. First

Usenix Workshop on Large-scale Exploits and Emergent Threats, no. 10, 2008; https://www.usenix.org/legacy/events/leet08/tech/full_papers/kanich/kanich.pdf.

4. P. Baecher et al., “The Nepenthes Platform: An Efficient Approach to Collect Malware,” *Proc. 9th Recent Advances in Intrusion Detection Conf.*, 2006, pp. 165-184.
5. L. Spitzner, *Honeypots: Tracking Hackers*, Addison-Wesley, 2002.
6. Y. M. Wang, “Strider HoneyMonkeys: Active Client-Side Honeypots for Finding Web Sites That Exploit Browser Vulnerabilities,” *Works in Progress: 14th Usenix Security Symp.*, 2007; www.usenix.org/event/sec05/wips/wang.pdf.
7. Y. Alosefer and O. Rana, “Honeyware: A Web-Based Low Interaction Client Honeypot,” *Proc. 3rd Int’l Conf. Software Testing, Verification, and Validation Workshops* (ICSTW 10), 2010, pp. 410-417.
8. N. Idika and A.P. Mathur, “A Survey of Malware Detection Techniques,” CS Dept., Purdue Univ., 2007; www.serc.net/system/files/SERC-TR-286.pdf.
9. “On-demand Detection of Malicious Software,” AV-Comparatives Lab, 2012; www.av-comparatives.org/images/docs/avc_fdt_201203_en.pdf.
10. S. Chamotra et al., “Data Diversity of a Distributed Honey-Net-Based Malware Collection System,” *Proc. Int’l Conf. Emerging Trends in Networks and Computer Comm.* (ETNCC 11), 2011, pp.125-129.

***Yin-Dar Lin** is a professor in the Department of Computer Science at National Chiao Tung University, Hsinchu, Taiwan, and founder and director of the university’s Network Benchmarking Lab and Embedded Benchmarking Lab. His research interests include the design, analysis, implementation, and benchmarking of network protocols and algorithms; quality of service; network security; deep-packet inspection; P2P networking; and embedded hardware/software codesign. Lin received a PhD in computer science from the University of California, Los Angeles. He is an IEEE Fellow and on the editorial boards of* IEEE Transactions on Computers, Computer, IEEE Wireless Communications, IEEE Network, IEEE Communications Magazine—Network Testing Series, IEEE Communications Surveys and Tutorials, IEEE Communications Letters, Computer Communications, Computer Networks, *and* IEICE Transactions on Information and Systems. *Contact him at ydlin@cs.nctu.edu.tw or through www.cs.nctu.edu.tw/~ydlin.*

***Chia-Yin Lee** is a postdoctoral researcher in the National Chiao Tung University’s Information & Communication Technology Laboratories. His research interests include cryptography, network security, and image processing. Lee received a PhD in computer science from the National Chung Cheng University, Chiayi, Taiwan. He is a member of the IEEE. Contact him at neko@nctu.edu.tw.*

***Yu-Sung Wu** is an assistant professor in the Department of Computer Science at National Chiao Tung University, where he leads the Laboratory of Security and Systems. His research interests include security, dependability, and systems. Wu received a PhD in electrical and computer Engineering from Purdue University. He is a member of IEEE and ACM. Contact him at ysw@cs.nctu.edu.tw or through www.cs.nctu.edu.tw/~ysw.*

***Pei-Hsiu Ho** is a postdoctoral researcher in the Department of Computer Science at National Chiao Tung University. Her current research interests include cryptographic protocols and mobile security. Ho received a PhD in computer science from National Sun Yat-sen University, Kaohsiung, Taiwan. Contact her at peyhsiu@gmail.com.*

***Fu-Yu Wang** is a project manager in the National Chiao Tung University’s Network Benchmarking Laboratory. His research interests include network and mobile security. Wang received an MS in computer science and information engineering from the Chung-Hua University, Hsinchu, Taiwan. Contact him at sagual@nbl.org.tw.*

***Yi-Lang Tsai** is researcher in the National Center for High-Performance Computing, founder and director of Cloud Security Alliance Taiwan Chapter, leader of The Honeynet Project, Taiwan Chapter, and leader of the Information Security Incident Response Team to handle security incidents for the Taiwan Academic Network (TANet) and Taiwan Advanced Research & Education Network (TWAREN). His research interests include the honeypot-related technologies and cloud security technologies for industry, government, and academia. Tsai received an MS in electrical engineering from National Cheng Kung University, Tainan, Taiwan. He has published 33 books and many columns in professional IT publications. Contact him at yilang@nchc.narl.org.tw.*

IT Professional
TECHNOLOGY SOLUTIONS FOR THE ENTERPRISE
www.computer.org/itpro

cn **Selected CS articles and columns are available for free at http://ComputingNow.computer.org.**